# INTERNSHIP REPORT

# DIGITAL SOLUTIONS FOR PAYMENT SYSTEMS

# SOFTWARE ENGINEERING INTERN

**Emirhan Delican**

**27820**

**Company:** **Interbank Card Center**

**Internship Supervisor:** **Feyzullah Orcun Cetin**

**Internship Start and End Dates:** **04/07/2022 – 01/09/2022**

**02/10/2022**

**SABANCI UNIVERSITY**

**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**COMPUTER SCIENCE AND ENGINEERING**

**Summer/2022**

# Abstract

This report will explain an internship process at Interbank Card Center (BKM). BKM is the heart of the payment systems in Turkey. It regulates the banking sector and develops innovative solutions to be used in payment technologies. To have a similar purpose with the BKM, the project that has been conducted during the internship was also designed to provide digital solutions for payment industry. The project was to develop a full stack online payment application, which has its own ecosystem to simulate all of the functionalities of a real world payment applications like “PayPal”. In order to build a structure close to the real world, some of the programmatic flows of BKM Express were examined and interpreted during the development of the project. The end products were satisfying with respect to the initial goals. A backend application, cross-platform mobile application and a web application were the final products of the project. Furthermore, the overall internship experience was pretty good. People were nice and wise. A suggestion is that it is important for intern candidates to be prepared for the possible duties to improve the quality of their internship. Therefore, they should search the working area of their company, and have an idea about the solutions that their company offers beforehand.

# Table of Contents

# 1 Introduction

I have conducted my internship at Interbank Card Center (BKM) as a software engineering intern. During my internship, I spent most of my time developing my project which is an online payment application that allows customers to make their online payments in a fast and secure way. The reason I decided to develop a project in this area is that it is very trendy, and demand for these kind of applications is very high nowadays. Also, it is closely related with my department “Bex” (Abbreviation of BKM Express) at the company. Lately, you may have noticed that nearly all of  big e-commerce enterprises has developed their own wallet system, which allows customers to deposit beforehand and spend in the future. In this way, customers will not have to write down their credit card information every time, which is considered to be insecure and waste of time. Actually, solving these problems (security and time efficiency) is the biggest motivation behind the online payment systems. They try to keep the user experience at high level by providing fast and secure money transferring methods. Although my application has way more general use case scenarios than this, I gave this e-commerce example to reflect the high demand to digital wallets and explain the importance of them. Similarly, the application I developed maintains its own wallet system. However, the functionalities are different from the example I gave above, as it is intended to be not company specific but a more general solution. For instance, it can work as a third party payment tool between a user and a merchant company. On the other hand, it also allows users to transfer money among themselves, so the receiver does not necessarily to be a company. Shortly, it is more like an abstraction of a banking system, but without the complexity. More functionalities and details about the project will be discussed in the upcoming sections of this report aside with a brief information about ICC, the details of the implementation, the internship experience I had and recommendations I make for the internship candidates.

## 2 Company Information

Company Name: Interbank Card Center (BKM)

Headquarters: Kultur, Nispetiye Cd Akmerkez D:E3 Blok, 34337 Besiktas / Istanbul

Contact Phone: (0212) 350 79 00

Web Page: https://bkm.com.tr

Number of Employees: 177

"The Interbank Card Center (BKM) was established in 1990 with the partnership of 13 public and private Turkish banks for the purpose of providing solutions to the common problems and developing the rules and standards of credit and debit cards in Turkey, within the card payment system." BKM is the heart of the banking system in Turkey from the day it was established. It provides tools that is a must to use for banks to keep functioning. As can be extracted from these details, BKM does not have a competitor, and although it is a private company, its primary goal is not making a profit directly, but to help the banking and payment systems in Turkey to keep progress. Furthermore, since BKM holds very crucial data for Turkey, the company has to follow the regulations of government that is sometimes only designed for them. Eventually, for various reasons the Central Bank of the Turkish Republic has bought %51 of the company in 2020. Since then, the management, engineer and human resources staffs has changed drastically. Therefore, it is safe to say that the company is restructuring, and the future goals of the company may change when this restructuring process is completed. This was the overview of the BKM.

As mentioned earlier BKM has many products that is essential for banking sector to exist, and payment systems to progress. However, it might not be possible for regular people to see these products in action, as they are mostly used in the background tasks by companies. "BKM Express" is the only product of BKM that interacts with the regular users directly. It is

the first digital wallet of Turkey which allows people to make payments at various merchant companies without entering their credit card information. However, it does not work as most wallets, it does not have a system where users can deposit money first and use it in future transactions. Instead, users attach their credit cards, and use them in any transaction that will be made through BKM Express. Furthermore, rest of the products of BKM are invisible for ordinary citizens, as they work like a protocol in banking ecosystem. One of the most important one among these products is called “Switch”. Switch is a system that makes the interbank transactions possible. Without it, each bank would only be able to handle the transactions that is happening in their own system. Considering the number of banks in Turkey, it is possible to say that without “Switch”, banking ecosystem would have collapsed in Turkey. Another important product of BKM is “troy”. Normally, every bank that is functioning globally uses the infrastructure of “Mastercard” or “Visa”. These two are standardized payment methodologies in banking ecosystem, as they provide a service in very wide range. Troy also provides an enhanced technological payment infrastructure for banks, but its range is currently limited within the nation. In the light of this information, “troy” may seem unnecessary, but it is a must. In case of an international conflict, Visa and Mastercard may decide to stop their services in Turkey. In this scenario, the domestic economy in Turkey would also be harmed very badly, if there were no such payment infrastructure as “troy”. These were some of the products of BKM.

There are 3 main departments in BKM which are divided into sections in their own (see Appendix 1). As all other Human Resources departments, BKM’s HR department is responsible for recruitment, performance evaluation and organizing educational programs that keeps employees up to date in their field. Moreover, the Research and Development (ARGE) department is where the products are developed according to the needs of the customer. ARGE has many sub departments, each controlling its own specific product. So that, every

product of BKM has its own core development crew that keeps the product in action. Also, ARGE uses Agile methodologies -mostly Scrum- in project management. Finally, Product Management department is the bridge between the ARGE department and customers. Their job is to make sure the product satisfies the expectations of the customer.

# 3 Project Background

## 3.1 Department Information

As briefly being discussed in the last section, ARGE is the department where the products are developed in BKM, and it was the only department that was related with the project I have done during my internship. It consists full of software developers, software architects, testers, system administrators, data scientists. Instead of having a duty based classification, ARGE has product based distribution of workers. So that, most of the time each worker only has to know the detailed implementation of the product that s/he is in its development team. The crew I had the chance to work with was “BKM Express” team. BKM Express is a digital wallet application that offers solutions in payment systems area, and it was one of the inspiration sources behind my project. Furthermore, since “BKM Express” is an old product, there was not much development tasks to be done. Instead, there were tasks about bug fixing, merchant integrations and sdk developments. There were 5 people in the BKM Express crew whom I closely worked with. Names, emails, and titles of those people are listed in the table below.

| Name | Title | Email |
|---|---|---|
| Kadir Guzel | Senior Software Architect | Kadir.Guzel@bkm.com.tr |
| Fatih Mehmet Tekcan | Software Architect | Fatih.Tekcan@bkm.com.tr |
| Ahmet Hamdi Cengiz | Junior Software Developer | Ahmet.Cengiz@bkm.com.tr |

| | | |
|---|---|---|
| Ersin Deniz | Senior Software Developer | Ersin.Deniz@bkm.com.tr |
| Gulcin Buyukbalci Altintop | Test Engineer | Gulcin.Altintop@bkm.com.tr |

## 3.2 Status of the Project at the Beginning

At the beginning of the project, the very first thing to do was deciding on tools, methodologies, and technologies to be used throughout the project, as I was starting from scratch. Luckily, there was no business side of the project, as it was only designed to mock an online payment system, so that I could focus on learning completely. Other than this, I needed to have an idea about the flow of a sample digital wallet that has functionalities which I intended to implement. Therefore, I examined the technology stack of BKM Express and tried to copy some of the flows it has. However, most of the flows were required to have many out of context applications for testing. So that, if I want to copy one of these flows into my project, I had to implement many other unrelated applications. In real life scenario, this would not be the case. For instance, “BKM Express” was only responsible for integrating their functionalities with their service providers, and developing necessary tools for their customers, whereas I needed to develop all three sides alone. Nevertheless, by deciding on what to do about these issues, I got myself a roadmap and proceeded accordingly.

## 3.3 Motivation

In the age of the internet almost every process handled more efficiently than before -in this context, efficiency refers to least cost with maximum output-. The internet was such an easy tool to use for people to handle their requests, it became a must for companies to exist in the internet ecosystem, as they would not survive if they do not meet the demand. Considering these information, it is not surprising that banking and payment industry has adapted their systems to internet, as payments are one of the most common routines in human life. Banks

leading the way, a lot of innovative payment solutions were developed. Digital wallets were one of the most famous of them. Besides the ease of use, they were also more secure than the traditional payment methods, as all of the processes were handled digitally without a human touch which reduces the mistake probability. In the light of these information, it can be extracted that the main motivation behind the digital wallets is to make the payment process easiest and safest as possible for people and companies. There are two common models of digital wallets. One of them is a company specific model that can handle the payments that are incoming to a company. The only purpose of this model is to serve its own company, whereas the other model is more like a third party aggregator which collects the payments for any company that is contracted with it. Besides the other benefits, the second model also reduces the expenses for a company to develop a payment infrastructure. The project I developed is based on the second model. There are three main features in my project which are listed below. A quick note is that all of the data that was used in this project was mock, but close to real world data.

1) Wallet system, where users can deposit, withdraw, and make payments.

2) Merchant system, where merchant companies will receive payments over my application.

3) Inner transaction system, where users can send money to each other.

## 3.4 Related Literature

Developing a real online payment application would have require many legal permissions, integrations, and security precautions. However, since the project I did has its own ecosystem, thus contains a mock data, I did not have to worry about any of these. For this reason, I only focused on the software development, and tried to make the flows as similar as it would have been in a real world project.

First of all, every interaction with the internet is handled over protocols. Http is one of the most common protocols that people use without being aware of its existence. It provides two way communication between a client and the server. When a user tries to reach a particular information from a website or mobile application, s/he sends a http request to that server. Server evaluates this http request and answers back with a http response. Although it was an oversimplified summary, it will be enough to understand the “REST API” methodology which was being used on the server side of the project I did. An API can be defined as predefined rules for two applications to transfer information with each other, and REST API is one of its implementations which uses http methods “post”, “get”, “update”, “delete” to handle database actions “create”, “read”, “update”, and “delete” (Dorairajan, 2022). With this REST API implementation on the server side, my project would be able to handle the http requests that is coming from clients. I used one of the most popular java frameworks “Spring”, specifically “Spring Boot”, to develop such a structure. After developed an API, next thing to do was to provide an interface for the users, so that they can use the app. This interface will be the client that sends a request to the API to retrieve, update, or delete a data, and when it receives the response, it will update the view accordingly. For this purpose, it is a standard to use “HTML”, “CSS” and “JavaScript” to develop an interface on the web. The job of HTML is to provide the basic structure of a website, whereas CSS handles the visual design, and JavaScript controls the HTML elements with respect to user events (Cox, 2020). Although there are many popular JavaScript frameworks that can do the same job alone, it is still required to know the basics of these three, as all of the code converted into html independent from the framework that has been used. I used ReactJS to develop a web client, as it has the biggest community support behind it. Moreover, the web client is necessary, but not enough for an online payment application. The main client that is a must to be developed for an online payment system is the mobile application since it offers

more practical and personal usage than the web client. Mobile application development does not have any standardized methods as web applications, there are many different options to build one. These options can be categorized under two sections “Native” and “Cross-platform”. Native applications are specific to an operating system, whereas the cross-platform applications can work on multiple operating systems. For instance, if a mobile application developed using “Swift”, which is a native development tool, only the mobile phones that has an IOS operating system can execute it. However, if that application was developed using a cross-platform development tool, it could have been executed by both IOS and Android operating systems. Each has its own advantage over the other, native applications offers smoother experience for the end user, but it is more expensive and time consuming to develop two separate native applications. Usually, big companies prefer to have native mobile applications, whereas startups and small companies prefer cross-platform applications. In my case, I preferred to use Flutter, which is a cross-platform development tool, as learning two separate technologies would consume a significant amount of my time at the company. This was an overview of the methodologies I have used in my project. More technical details will be discussed in the upcoming section.

# 4 Internship Project

## 4.1 Project Objective

As explained in the earlier sections of this report briefly, the payment application I developed has its own ecosystem. It only mimics the characteristics and functionalities of the real online payment applications. Thus, it does not have any effect on real world. Although, this provides a convenience from many aspects, it also makes the development process much harder. The reason for that is, in this path you need to replicate numerous of tools that banking ecosystem already has for years. For instance, the application I developed accepts credit cards

as a payment method, in a real world scenario the only thing that is needed to be done to accomplish it is to make the appropriate integrations with the banking ecosystem. However, in my case there is no built-in banking ecosystem, so that the card could be validated. Therefore, I had to develop a simple banking infrastructure, even though it is not related with my project. Moreover, within the scope of the project, I developed a sample merchant web application, which accepts payments through my application. The client, server, database flows of this functionality were directly copied from the flows of "BKM Express". Although, there are various software flows used in the project that is close to real world applications', the most common security practices, such as "Oauth", "JWT", were not included in the scope of the project. Instead, a simple access token system was used for security measures. This was to give an insight about the scope of the project. In a nutshell, by providing various solutions in payment processes, in general money transferring, this application simulates how an online payment application improves the experiences of its users.

## 4.2 My Responsibilities

The project I have conducted during my internship was an individual project. I did get help from my crew time to time, but all off the designs and implementations were done by me. The first step of the project was learning every concept that I planned to include the project. For this purpose, I have read many documentations, and watched tutorials. While learning a particular technology, I also applied it into my project. Also, whenever I saw a suitable technology that solves a particular problem in my project, I did not hesitate to dive into it. Actually, many technologies were added into the project in this way. For instance, in most resources microservice architecture was explained using Docker and Kubernetes examples, so that I decided to include them into my project. That was how I proceed in the development of the project.

## 4.3 Methodology

In the scope of this project, I used Spring Boot, a java framework, to develop a microservice architecture, to containerize these microservice applications I used Docker, then I deployed these containerized microservices into a Kubernetes cluster. This was the backend side of the application I was going to develop. On the frontend side, I used ReactJS to create a web application, and Flutter to create cross-platform mobile applications. This was the summary of the tools I have used, the detailed explanations about how they were used inside the project will be made in the upcoming sections.

## 4.4 Expected Outcome

The expected outcome of this project is a full-stack online payment application that mocks the functionalities of the real world payment applications. This is going to be achieved by using the methodologies mentioned in the previous section.

## 4.5 Details

### 4.5.1 Overview of Backend

As you may recognize from the previous sections, the backend side of the project was developed upon a microservice architecture. In traditional way of developing a backend, there is only one application that is responsible for all of the functionalities of the application. However, this approach is problematic from various aspects. Scalability issues, single point of failure risk, maintainability are few examples of these problems. Microservice architecture solves most of these problems by separating the business logics of each functionality and creating a unique application for each. The tool that was used to build these unique applications was “Spring Framework” of Java. The first step of creating a microservice using Spring Boot is to select the dependencies of your application from “Spring Initializr”, and

downloading the codebase generated by it. Then, dependencies can be managed through "pom.xml" file for further updates. In my case, most of the microservices had the same dependencies which are "Spring Web", "Lombok", "Spring Data JPA" and "PostreSQL Driver". Before explaining each, there is another configuration needed to be done before the development starts, the database server configuration. As you may have guessed from the dependencies, the main database used in this project is PostgreSQL, which is a relational database. In order to use PostgreSQL in your application, you need to setup the environment accordingly. I used Docker, which will be furtherly discussed, to set a local PostgreSQL environment on my computer. After setting the environment, I configured the "application.properties" file to complete the integration between my application and local PostgreSQL database. A quick note is that Spring Boot handles the complex integration between the application and database server with its auto-configuration features coming from the dependencies, so that the only thing to do as a developer is to provide the address of database and admin credentials. Although the best practice is to use different databases for different microservices, I preferred to use only one database for entire application, considering the scope of the project, it would be an overkill otherwise. At this point, I would like to explain the "Spring Data JPA". In order to interact with the database on the backend side, developers need data access layers. Most of the time, developing a data access layer from scratch is an overwhelming process. Spring Data JPA solves this problem by providing the implementations of the functionalities that developers need. The only job for developer is to create a repository interfaces, and add custom methods inside to it, the implementation will be handled by Spring Data JPA. Spring Data JPA uses Hibernate as an Object Relational Mapping (ORM) tool. Hibernate is an ORM framework that makes an appropriate translation of object oriented models into the database (see Appendix). In other words, developers do not need to write a single line of SQL, if they are using Spring Data JPA, as the Hibernate handles

all of the necessary SQL commands that is instructed to it by the developer. Developers give these instructions in different ways for different cases. For instance, creating a method inside an interface that extends “JpaRepository”, will basically takes the place of SQL queries. Another example is using annotations for creating and modifying a table in the database. Annotations that Spring Boot provides are super powerful and easy to use, thus it is important to get familiar with them. For instance, “@Entity” annotation that is placed before a class definition, sends a signal to hibernate to generate a SQL statement that creates an entity in the database according to the attributes of the class. “@Bean”, “@Service”, or “@Repository” annotations that is placed before a class definition, makes that class singleton. Singleton is a creational design pattern, where each class can have only one instance during the runtime. Last example is “@Autowired” that handles the dependency injection, which means finding and wiring an existing object into the newly created object. These were some examples of annotations that Spring Boot provides. Another annotation provider is “Lombok”, which its dependency was added into the project at the beginning. It provides the most basic needs of a class with annotations, such as getters, setters, constructors. These were common characteristics that each of the microservices share.

### 4.5.2 Docker

Docker was used for two different purposes in the project. First of them is to set the development environment properly, and the second one is to get the image of the microservices. Setting the development environment with docker, is more secure way compared to download each of the dependencies alone, as their versions might not be working well with your operating system. Docker removes this probability of compatibility issues, by running the applications inside of a container that has its own ecosystem inside it. In other words, if an application can run in one environment, when you containerize it with docker it can run everywhere without compatibility issues. For the same reason, it makes sense to run

microservices inside containers, as they will be deployed somewhere else other than your working environment in real life. In order to containerize the microservices developed using Java, the first step is to create an executable of the program. After that a “Dockerfile” is needed to be created inside the base directory of the current project. This file gives the necessary details about how the containerization should be done. For instance, it contains the “base image”, executable and entry point information. The executable is written in some language, but in order to run this executable there should be a mechanism to understand what it says, this is the base image information inside the Dockerfile. Similarly, entry point tells what should be started during the initialization of the container, which is the executable that was created beforehand. These were the general use cases of Docker inside the project.

### 4.5.3 Kubernetes

Kubernetes has very detailed and complex structure, therefore the technical details of the Kubernetes will not be discussed in this report. The containerized microservice applications were deployed inside a local Kubernetes cluster. In order to do that I prepared a Kubernetes manifest file in the form of yaml for each of the microservices, then applied these files using command-line tool of Kubernetes “kubectl”. After this point, for each of the microservices, a pod is created by Kubernetes. However, these pods are not guaranteed to live forever, they may die. This is not a problem when you are using Kubernetes since another pod will be instantly replaced with the dead pod. On the other hand, newly created pod will not have the same IP address with the dead one. This is a very problematic scenario for microservice applications that is exposed to real world, since the known IP address is not valid anymore. In other words, having a dynamic IP address is a problem for a particular microservice application. This situation brings up the service discovery concept. According to these concept, all of the microservices have to register themselves to the service provider, so that the service provider will be aware of the changes happening. In this way, when the dead

pod replaced with another pod, service discoverer will notice the change and update the IP address of that particular pod. Service discovery is done by Kubernetes Services that is specific to each deployment. The best practice is to have a different service for different microservice deployments. Besides the issue discussed above, overloading a single pod will also result in connection problems. Therefore, it is a good practice to multiply the same microservice application inside another pod. This is an example of horizontal scaling, which is one of the key motivation behind the emergence of microservice architecture. Notice that the newly created pod will also register itself to service discovery, in this case when a new request comes into the service, it should choose one of the pods to redirect the request. Luckily, Kubernetes services can also act as a load balancer, so that each request is redirected to a pod with respect to round robin. Moreover, exposing all of the Kubernetes services to outside world is not safe and practical. Therefore, it is a good practice to have an only one entry point inside the cluster, which is called “API Gateway”. The requests that come to the API Gateway, are redirected to related microservices using the endpoints of the requests. These mappings between endpoints and microservices should be described in a Kubernetes Ingress file. All of the practices of Kubernetes explained above were used inside the project to gain a practical experience, although it does not make any sense to use them in local environment.

### 4.5.4 Microservices and Their Duties

**User Microservice (UMS):** It is responsible for actions like registering, logging in-out, updating balances, validating an inner transaction, getting user details etc. For instance, when a user tries to login, a post request comes into the endpoint “api/v1/users/login”, then it is checked if the credentials are true and respond back accordingly.

**Credit Card Microservice (CCMS):** It is responsible for validating and storing the credit card information that was entered by the user. It communicates with the Credit Card Database

Microservice (CDMS) in order to validate a card. For instance, when a user tries to attach a credit card into its account, a post request comes into the endpoint “api/v1/cards”, then the CCMS sends a post request with the credit card data inside the body to CDMS, CDMS checks from the database if the credit card exists in the database and responds accordingly. If the response contains a success message, CCMS saves the card in the database, and responds back with a success message to the client.

**Credit Card Database Microservice (CDMS):** It is responsible for validating credit card data, checking balances, money transferring among credit cards. Basically, it is the representation of the banking environment of real world. For instance, when a user wants to make a payment to a merchant company with its credit card registered on the system, PMS will send a post request to the CDMS, CDMS will check if the credit card exists and it has a sufficient balance, if both conditions are met, it will transfer the amount from one account to another.

**Deposit Microservice (DMS):** It handles the incoming deposits to the system. For instance, when a user tries to deposit money with its credit card, as usual, the credit card data will be checked by communicating with CDMS, then DMS will send a post request to UMS to update the user balance, if it succeeds the deposit information will be saved in the database.

**Transaction Microservice (TMS):** It handles the money transfers that is happening inside the system. For instance, when a user wants to send money to another user, TMS will receive a post request, then it will communicate with the UMS using http methods to check if the sender has enough balance, if it has UMS will update the balances of both accordingly and responds with a success message. After that TMS will save the transaction information in the database and response back with a success message to the client.

**Payment Microservice (PMS):** It handles the payment processes. For instance, assume that a user is shopping online, and that company accepts payments through this application. So that

its virtual pos information, and other necessary stuff are registered in the system. When the user selects this option to complete the payment PMS receives a post request with the details inside the body, it creates a token for this payment, saves it in the database, and respond back with a token. When the client receives this token, it redirects the user to login screen of this application. It also includes the token as a path parameter to the path of the login page. So that, when the user logs in, client sends a get request to PMS with respect to the path parameter, to get the details of the payment. Then, user selects a credit card to complete the payment. After clicking the “complete payment” button, necessary checks will be made among PMS and CDMS, if nothing is wrong PMS will return success message.

**Mail Microservice (MMS):** It is responsible for sending mails to the user after some events. For instance, when a user registers UMS generates a token for it and sends a post request to the MMS with the token and registered mail information on the body. When MMS receives this request, it sends a confirmation mail to the user with an endpoint similar to “…/validate?confirmationToken=xxxxxx-xxxx-xxxx-xxx”. When the user clicks this link, UMS receives a get request, it finds the token from the database and activates the account of the user that the token belongs to.

**Session Id Microservice (SIMS):** It is responsible for authorization. When a user logs in, an access token will be generated and attached to its session by SIMS. In every request of that user, the related microservice will communicate with SIMS to validate the user has permissions to do that action. Every access token has 5 minutes before the expiration, after five minutes when the user makes a new request, the access token will be deleted from the database and user will be forced to log-out on the client side.

### 4.5.5 General Structure of Microservice Applications

In this kind of medium sized projects, it is crucial to keep everything maintainable, so that the development process can continue forever. Therefore, it is important to write a clean code with efficient file structure. I have used three layered structure while developing the project. The names and duties of these layers are listed below.

**Controller Layer:** Controller layer is where I created the endpoints that receives the http requests. For instance, the controller class of the User Microservice has an endpoint "…/api/v1/users/login" attached to a method, so that whenever a post request comes to this endpoint, the attached method runs. However, the only job of this method is to forward the necessary information, such as path parameters, path variables, headers, to the service layer.

**Service Layer:** Service layer is where the business logic is handled. It communicates with other microservices, interacts with the database through the singleton data access object, throws an appropriate error if necessary.

**Data Access Layer:** Data access layer is where the interaction with the database is handled. Since the "Spring Data JPA" was used in the project, The only thing to do in this part was to create an interface and add the necessary methods inside it. For instance, when you want to find a user with its id, you create the prototype of the function inside the interface as " User findUserById(Long id)", the implementation of this method automatically created by Spring Data JPA.

Besides these three layers, there are also two more side concepts, which are data transfer objects (DTO) and entity classes. Entity classes are where the database entities were designed with the help of annotations. For instance, if you would like a class to mapped into an entity in the database you need to put "@Entity" annotation above the class. It is also possible to manage primary keys, foreign keys, nullability and many more with the annotations. Moreover, the DTO's are necessary to prevent a security breach, when

responding back to outside world. For instance, if an entity class has an attribute that should not be seen by anyone, then the response cannot be an instance of that entity class. There should be another class without that attribute, and the object to be returned should be an instance of this class. This was the general structure of the microservice applications developed in this project.

### 4.5.6 Frontend

The frontend part of the project is not detailed as the backend, it is only designed for testing the functionalities of the backend, therefore it is not based on a proper UI/UX design. As mentioned earlier, ReactJS was used to develop the web application. For state management and routing, no external libraries were used. On the mobile side, Flutter was used to develop a cross-platform mobile application. For state management and routing, Getx was used. Both of these applications, interacts with the Web API designed by me. For instance, there is a login view in both of these applications where users can enter their login credentials. Then when they press the login button, the credentials are sent to the API Gateway inside the body of a http post request. By looking at the endpoint-microservice mappings, Kubernetes redirects this request to the responsible microservice. When microservice receives this request, it runs the responsible method for this request. Then it gives a response to the client. When the client receives the response, it updates the view accordingly. If login succeeds, the dashboard page will come to top of the view stack, if it fails a proper message will be shown on the screen. Shortly, you can imagine that for every functionality that is developed on the backend side and needs an interaction with the user, there is a page on the frontend to make the interaction possible.

### 4.6 Results

By comparing the end products with the initial purposes, it is safe to say that the project has been completed. However, this does not mean that it has the ultimate functionalities, and no development process needed. It only covers the basic functionalities that an online payment applications could have. For this reason, many other unique features can be added further. Moreover, since the project does not have a connection with the real world, it does not make sense to put it into action. It is useful for testing and demonstrating purposes. Also, it would be very costly to deploy all of these applications, considering it will not make any profit.

## 5 Internship Experience

### 5.1 Learning

The very first thing that comes to my mind when I ask myself "What did this internship taught me?" is the ability to learn and use new technologies. Of course, I gained a lot of practical experience and software related skills, but I believe that these can be obtained anytime in life if you know how. What I mean is that instead of focusing on learning new technologies without a proper approach, you should focus on the questions like "how should I learn this, in which way I can learn this technology better, do I already know the prerequisites of this technology…". By answering these questions, you will get yourself a roadmap to proceed. I believe this was the most valuable ability I gained during my internship "knowing how to learn".

By evaluating my internship experience on my own, I am convinced that I should be a software developer, because I enjoyed my time at the company developing different stuff with different methodologies. On the other hand, the internship gave me an idea about how painful

it might get working at banking sector, so I'm not so sure if I should work at banking (payment systems in general) sector. The future will show if it is possible, since the banking sector has a large share in IT industry.

## 5.2 Relation to Undergraduate Education

During my internship I came across many concepts which I could recognize from the university courses. However, before I explain them, I need to mention a more general ability I used during my internship “Problem Solving”. Problem solving is a prerequisite of almost all programming concepts. Without this ability, no matter the effort you put in, you will fail to get the objective done. The problem solving skills I gained from the IF100, CS201, CS204 (starter CS courses at SU) was more than adequate to accomplish any task I engaged during my internship. Furthermore, as a requirement of my project I designed and used a SQL database. Even though I used Spring Data JPA, which handles SQL statements automatically using ORM tool “hibernate”, still I had to be familiar with the traits of SQL based database systems like “foreign keys”, “atomicity”, ”one to many”, “many to one”. Thanks to Database Systems (CS306) course my SQL knowledge was way more than sufficient to meet these requirements. Another key feature of my project was a mobile client, which I developed using Flutter. This was a direct opportunity for me to practice concepts like “state management”, ”routing”, “coding a user interface”, which I learned from Mobile Computing (CS310) course. Although I did not use the same techniques that I have been taught, the logic was same most of the time.

Aside from the concepts that I could recognize from university courses, there were many new topics and areas that I was introduced. API development (Backend Engineering) and “deployment process of a project” (DevOps Engineering) were only two of them. Although these two concepts are like the foundation of the modern web, there is no university

courses related to these. I believe that these two give a very good insight about what you can do with all the things you have learned from the other courses and should have courses on their own.

## 5.3 Difficulties

The hardest part of the internship was setting up the working environment on the laptop that was given by the company. Due to security policies of the company, downloads were restricted, and required to have a lot of permissions. The process of getting required permissions was very slow, and I can easily say that it could take weeks. Since my time was limited at the company, I did not want to wait. Therefore, I started to use my own laptop for development, and used the other laptop for following the company related responsibilities. Since people were working remotely, there were very few people in the office at the beginning of my internship, which delayed my orientation a little. This was another difficulty I encountered during my internship, but luckily it did not take too long.

## 5.4 Typical Day

Although my internship was hybrid on the record, my company told me to do so, I went to work all 40 days. All of these working days were the same more or less. The working hours were flexible between 8am - 5pm and 9am – 6pm, I preferred the first one since there was less traffic during that hours. I was usually starting the day with a breakfast which was given by the company to all employees free of charge. Then until the lunch break, I was working on my internship project, I was asking questions when I was stuck. After the lunch, if there is a meeting, I was attending it as a listener. Sometimes my supervisor Kadir Guzel was explaining some concepts like “the flow of BKM Express”, “hashing mechanism, encryption, JWE” to me one to one. These one to one sessions were very effective and enlightening. For two times during my internship, I made a presentation about my project to whole team. I

explained them the technologies I used in my project, and they gave me a feedback. These sessions were also beneficial for them, as they were interested in containerization and how it is done in the background. That was pretty much my typical day at the company.

# 6 Conclusions

During my internship period, I have tried to build an application that mocks an online payment system. From my point of view, the project was three staged. These three stages can be summarized as developing and deploying an API that would be able to handle the backend communication, developing a web application and cross-platform mobile applications that offers an interface for the user. By doing all of these developments alone, I gained a general knowledge about software engineering and banking domain. Microservice architecture, Docker, Kubernetes, backend development, frontend development, mobile application development, relational-nonrelational databases, database caching, design patterns are the major software skills I practiced during the development of the project. Besides the software experience that I have gained by working on the project, I have also gained corporate experience by joining daily meetings, making code reviews, working as couples. To sum up, although the 40 working days is a very short time period to adapt the company and build a project, the internship experience I had was very enlightening in terms of my career plans. Also, considering the major purpose of an internship for the intern is to improve himself/herself, it is safe to say that this internship could not be more beneficial, as I improved myself in various fields.

# 7 Recommendations

At the beginning of my internship, I was ready to fulfill the requirements of any task that is assigned to me. The end product might not be perfect at that time, but at least I definitely had an idea about what should be done to complete the task. I believe that this

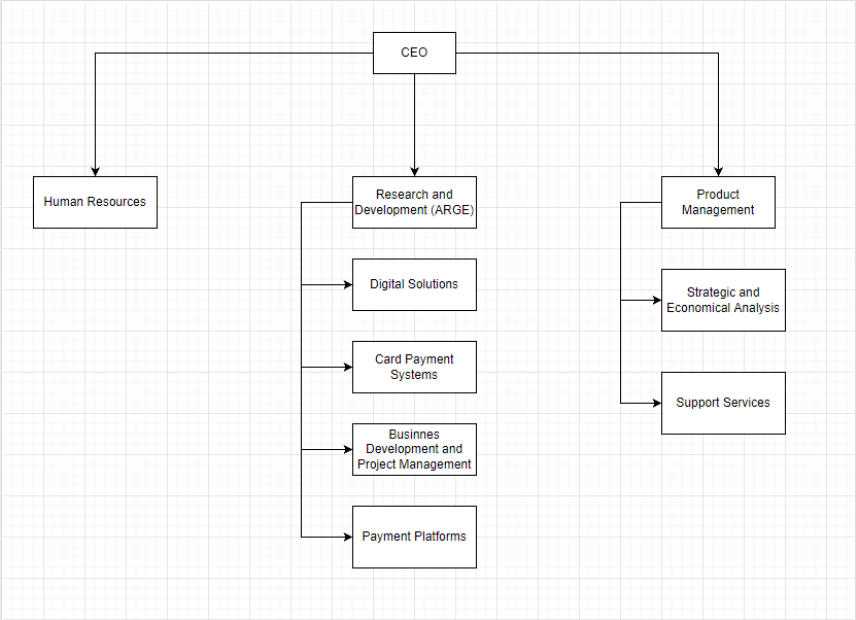
capability of mine gave me a self-confidence, which made me a person who willingly tries to take more and more responsibility. By extracting from my own experience above, I would recommend internship candidates to search for the working area of their company and have an idea about solutions that their company offers. By doing that they will not lose time, as they will understand and embrace the products of the company easily. Another recommendation I would like to make is that they should limit their expectations from the company and be prepared for the worst case scenario. In my own internship, luckily, this was not the case. The people at my department were wise and helpful. However, I can safely say that it is very likely to face with another path that might not be that good. For instance, if people are working remotely at your company, they might not bother to come to office to guide or even meet you. In this scenario you should be able to proceed alone, which is feasible if only you followed my first recommendation. Lastly, make a notes of your questions if it is not possible to ask your questions often. I am very regretful for not doing this, I would have improved myself in a more efficient way if I have done that.

# References

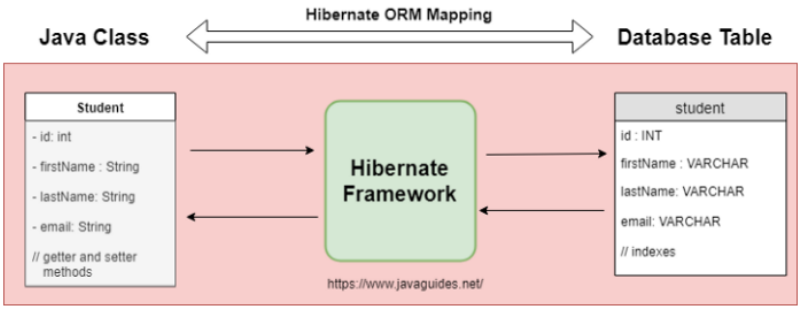
Cox, L. K. (2020). Web Design 101: How HTML, CSS, and JavaScript Work. Retrieved September 20, 2022, from https://blog.hubspot.com/marketing/web-design-html-css-javascript

Dorairajan, A. (2022). What is an API?. Retrieved September 20, 2022, from https://blog.axway.com/learning-center/apis/basics/what-is-an-api

# Appendices




*Appendix 2: Organizational Chart of BKM*




*Appendix 1: How hibernate works*