INDUSTRIAL TRAINING REPORT

TRAINING ORGANIZATION	: 20FACE B.V.
PERIOD OF TRAINING	: FROM 19/02/2019 TO 09/08/2019
FIELD OF SPECIALIZATION	: COMPUTER ENGINEERING

S.M.H.M. SAMARAKOON

E/14/302

ACKNOWLEDGMENTS

It is a great pleasure to undergo a well-planned and well audited internship for my bachelors in computer engineering. I am thankful for Prof. Roshan Ragel, Dr. Dhammika Elkaduwa, Dr. Isuru Nawnina and all the other lecturers of the Department of Computer Engineering.

Also I would like to extend my gratitude to the Head of the Industrial Training and Carrier Guidance Unit Mr.W.R.M.U. Wickramasinghe for providing the necessary documents and the support.

Next I would like to acknowledge Dr. Tuseef Ali -CTO at 20face and fellow employees who made this foreign internship possible.

I would like to thank Eng. Wout Oude Elferink - Technical Lead of the SDK development team for guiding and helping throughout the internship. Also I take this opportunity to thank Chris Boeren - Senior Software Developer for sharing valuable insights about software Development in a wide scope. These insights for sure will be a big benefit to craft my professional career.

Last but not least I would like to thank all the people who made this internship a success by guiding me and supporting me.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
ABI	Application Binary Interface
FLANN	Fast Library for Approximate Nearest Neighbors
GDPR	General Data Protection Regulation
JNI	Java Native Interface
PDRM	Personal Data Release Management
NDA	Non-Disclosure Agreement
SDK	Software Development Kit
RAM	Random Access Memory

Chapter 1

INTRODUCTION

1.1 TRAINING SESSION

I had my internship from 19th of February 2019 to 9th of August 2019 at 20face B.V which is located at Gallery 1.70, Hengelosestraat 500, 7521 AN Enschede, the Netherlands.

1.2 INTRODUCTION TO THE ORGANIZATION

1.2.1 What is 20face?

The name 20face originally comes from the word 'Twente' a province in the Netherlands. It is a startup company founded in late 2017 by a group of persons (Dr. Tueseef Ali, Prof. Raymond Veldhuis) who are related to the University of Twente. 20face is a spin-off from the University of Twente. It is growing to become a scale-up company [1].

What makes 20face special from other commercial facial recognition systems is by its privacy proof by design concept. 20face has approached handling biometric data securely. For the last two years 20face has been building a platform that enables privacy proof recognition in parallel to their main objective - researching on improving the facial recognition software. The personal data release management (PDRM) is one output 20face has been working on. The reasons behind 20face's approach for more secure system are to avoid dissatisfaction among users and businesses.

Since 20face is more concerned about middleware and not on the end customers 20face can also be identified as a B2B company.

1.2.2 20face Vision

The vision of 20fcce is to continue to build the facial recognition tools in a way that it is more scalable and secure. Also their vision is to sustain and compete with its emerging competitors in Europe. The prime objective of 20face is to integrate the facial recognition platform to all the applicable systems country wide.

20face aims to build its technology extremely scalable that its product work for millions of people in different applications. From the point of company level of view, it is thriving to move from startup to scale up level.

1.2.3 20face Ecosystem

20face has built an ecosystem [2] of self-enrollment, consent and personal data release management (PDRM) around the core technology of biometric recognition. To brief the self-enrollment process, the user enrolls him in the system through a mobile app. A photo is taken and uploaded via the app. 20face facial recognition tools that run on the system distils the face vector and encrypt the vector in a hash. This makes it impossible to reverse engineer the face vector. The photo is securely deleted afterwards. The hash and the face vector are distributed to the endpoints (can be football stadium gates, saloons, government organizations like DigiD) that the user has given consent to. The above process is in aligning to Personal Data Release Management (PDRM). That is only the user can consciously grant permission to be recognized at a particular endpoint. Also the user enables what type of biometric data to be shared at a given endpoint. For an example he or she may give access to facial recognition but age detection.

20face has a robust platform that is flexible and scalable. The company's goal is to provide custom solutions and act as an agent in the middle that connects end user and the endpoint. 20face uses Distributed Ledger Technology, the underlying technology for the block chain. This makes it virtually impossible to obtain facial data from users.

1.2.4 The Management Structure

20face has three departments, Administration, Research and Development. The researchers are well experienced in the field of facial recognition and the software developers are building the 20face platform. Apart from that there is an advisory panel that drives 20face to success. The overall hierarchy can be displayed as in the figure 1.1 below.

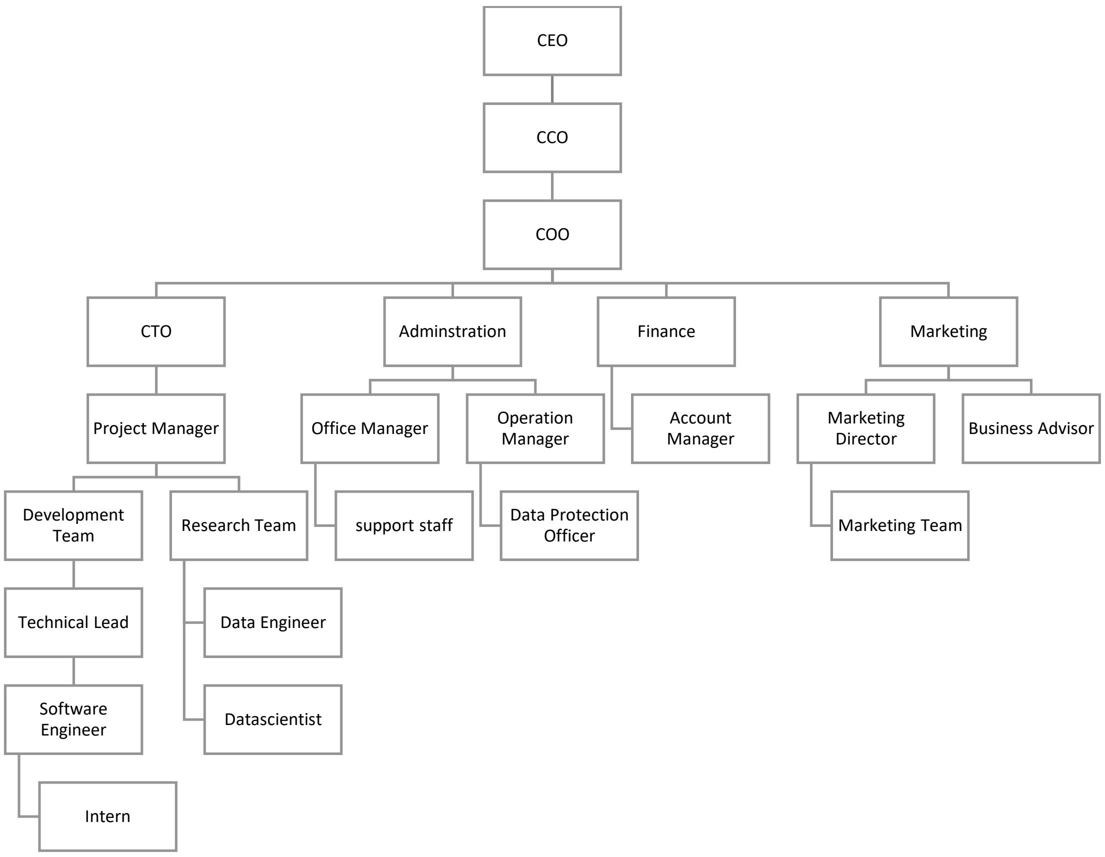

Figure 1.1 Organization Chart

1.3 SUMMARY OF TRAINING EXPOSURE

This training report is made with the experience gained during my internship at the company 20face. The report consists of four chapters. First chapter includes information about 20face, a brief history of the company, its vision and objectives and the management structure.

The second chapter is dedicated to illustrate the personal experience that I had during my stay at the company as an intern. Starting from the first day at the office until the last day the incidents including the brief orientation program are described in this chapter.

The third chapter is more technical. It describes the main project that I was assigned to work on. My project was to support the implementation of the SDK. This SDK is supposed to be the underlying system for the 20face platform. More details can be found in chapter 3.

Fourth and the last chapter illustrate a mini project that I worked on. It was to build a training platform to train the 20face face models. This is an emerging technology to use the cloud services to build and train machine learning models. The pipeline makes it easy to benchmark and deploy. More details can be read in chapter 4. It is noteworthy that chapter 3 and chapter 4 contain my personal comments on the tasks that I did. I have also included personal opinions about the company and the things I learned during the projects that are not technical.

Chapter 2
ORIENTATION PROGRAM

2.1 INTRODUCTION

This chapter covers the details about the first week and the last day at the office. The orientation program which was held during the first week (first three days) helped to get an overview of office and what my role will be.

2.2 FIRST DAY AT 20FACE

I was welcomed by the Account Manager at 20face, Catherine Ter Haar. She was actively working on Human Resources and Administration as well. Then I was introduced to the other colleagues in the office by the Chief Technical Officer, Dr. Tuseef Ali. It is worth mentioning that everyone addressed each other by their first names, instead of the posts they were holding. That helped to integrate into the working environment easily. I had the chance to talk to everyone in the office personally and to get to know what they were working on.

2.3 ORIENTATION PROGRAM

2.3.1 Integration to the Company Culture

Even though the number of people working in the company is small, they belong to a variety of cultures and nationalities. This made me to have an international experience to its fullest. I had the chance to share thoughts with Ducth, Spanish, German, Pakistani and Indian people. Above all I was introduced to the environment where they all had lunch together. This reduced the cost and time which can be used for other meaningful work. On Fridays the company ordered lunch from the outside.

One other thing about the working culture is that almost everyone in the office joined for a short walk after lunch. This helped to interact and blend with others that are from very different social backgrounds. I think it was my best opportunity to experience group dynamics.

20face was inside the building called Gallery. Gallery hosted a lot of startups. Everyone in the building shared the washrooms facilities. Gallery is located inside the University of Twente premises. Most of the startups were originated at the university.

Companies in the building had a lot of investors visiting them on a regular basis. Hence, it was very normal to see posters and banners outside every office, describing what the company is doing. Working at 20face, I got to know a lot of other startups as well. Building had silent areas which are accessible to any authorized person. I was given a key to the office, to the building and to the cycle park.

Gallery had a laboratory called Design Lab. Students who are following B.Sc. and M.Sc. degree programs were working on this laboratory. There was no boundary between employees and the students. It was exciting to see employees being busy with playing table tennis in the Design Lab in the lunch hour. At the same time students were always welcomed at the offices. Every first Monday of the month there was a security checkup carried out by the maintenance staff of the building.

2.3.2 Introductory Sessions

I was given an overview about the company on the second day. I have illustrated the company structure in the figure 1.1, according to the knowledge I gained in this introduction. I had two meetings on the third day. The first one was with the SDK development team. There we discussed what should be done in order to make the project a success. For more details about this please refer chapter 3. Then I was introduced to the research team. I was given a brief introduction about what the research team was doing and where 20face face model stands. The fact that 20face face model was in the first 25 models in the world, motivated me to start working. Nevertheless, since most of the face models are proprietary and hidden there was no ground truth that these models can be compared against.

There was a brief presentation about the technical culture, rules and regulations 20face follow for sustainable development. As a biometric processing company 20face is well aligned and compliant with GDPR (General Data Protection Regulation). It was emphasized how much 20face takes care about the privacy. 20face has collaborated with SMILO - a data protection company that is also a startup which uses block chain concepts to secure data. The higher degree of concern about privacy proof biometric data has lead 20face to win a big fund from the Dutch government.

After the awareness program about legal aspects an account was created under my name to access 20face resources. Using the account I was able to access Version Control System and its repositories. I could also access the Documentations and 20face’s communication platform. The introductory session was over after I signed the contract forms and particularly the NDA (Non-Disclosure Agreement).

2.3.3 Getting Hands on Technology Stack

I learned that 20face was using Bitbucket as the version control system. 20face used JIRA to plan the sprints and to maintain the backlog. Confluence pages were used for documentation. Since I had sufficient experience with using GitHub it was easy to get things started with Bitbucket. Nevertheless, I requested a short example demonstration of the life cycle of a small code snippet. Pablo Garcia who was another intern demonstrated the entire pipeline with a small example.

20face used SLACK for communication. There were different workspaces such as development, management, random etc. For an example development workspace was where developers exchange their ideas and links. Random space was fun and exciting with all the programmers' fun.

The programming language used in developing the SDK was C++. Even though it was not exactly what I have learned at the university, having the knowledge about the fundamentals helped me to quickly get myself comfortable handling the technology stack. The basic cycle of development is shown in the figure 2.1

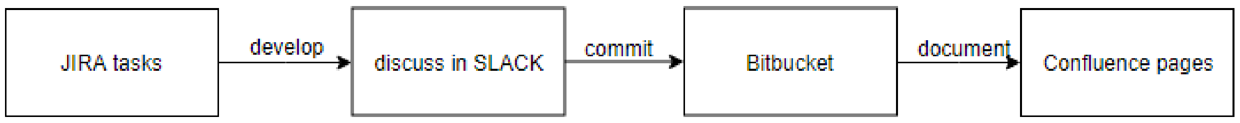

Figure 2.1 Development Cycle

2.4 JOINING THE DEVELOPMENT TEAM

As I have mentioned before, the development team was involved with their new project - 20face SDK. I officially started working on the project from the 22nd of February 2019. I was asked to go through the source code that was built so far and also to study the Class Diagram. I have learned C in the University. I have done C++ programming to some extent. Although I had no experience with Object Oriented Programming in C++, it was not a drawback. As I knew the fundamentals of Object Oriented Programming and Java it was easy to catch up.

I studied the Class Diagram and Use Cases very carefully. Then later in the day we had a meeting to decide on what part of the Class Hierarchy should I start working on. It was decided that I should start with implementing the Database Class. Chapter 3 has a more detailed view about the work.

2.5 EXPERIENCE GAINED

The orientation programs held at 20face helped me to improve my understanding about the professional environment. In the university the law enforcement regarding a project was not taken quite serious. Here, I learned that the whole enterprise can collapse with a violation of a small regulation. I learned how different departments work together to achieve one goal - success.

Also I understood how big an impact can be made by the relationships within departments. Moreover, I could perceive how important communication is. I had to communicate within the SDK development team, with the HR division and occasionally with the higher level management. This improved my confidence and soft skills.

Also I studied how 20face being a small startup has thrived so far along a course of two years. Engaging with colleagues at lunch time improved my insights about the rest of the world. Sharing culture among them lead us to look at the world in a completely different perspective.

I learned how to effectively communicate with fellow professionals, present the ideas and problem solving skills. Also how to use our Social media networks, blogs and open source contributions to build the career. After the orientation I was fully integrated into the social and technical environment. Starting from coding styles to documenting code everything was worth taking care of.

Chapter 3

MAIN PROJECT – BUILDING THE SDK

3.1 INTRODUCTION

I started working on a project where we had to build a commercial product, a Software Development Kit (SDK). SDK should have functions to invoke the facial recognition tools that were built by the research team. We started implementing the SDK from scratch using C++. After the project my knowledge about the software field leveled up. I learned the best practices in coding, interface design, optimization, continuous integration and many more. What made this learning process very interesting was that I had to use what I've learned in a real world scenario. Moreover, the fact that the commits that I make has a direct impact on the production and also affect the betterment of the company made me take full responsibility over my work.

3.2 20FACE SDK

20face SDK will be the building block of the 20face ecosystem. It was developed in such a way that the users can implement their own facial recognition systems on top of it. SDK enables integration into various different applications. Also the SDK communicates with SMILO APIs. SDK has the inference neural network of the 20face face model. The neural network is encrypted so that the SDK became sole proprietary. Users can use the functionality provided by the SDK. User can perform face detection, face enrollment, face recognition and face comparison as core functions. The SDK also had a set of supplementary functions that enabled video streaming and recognition on the fly.

20face SDK was a product of 20face. It was released to the users in versions. It was up to the user to decide whether to update the SDK when a new release was available or to keep the current version. The SDK came with a license. Users were advised to renew their licenses every six months. SDK was backward compatible and never broke the legacy functionality. That was a promise made by 20face to the customers. The ABI of the SDK will not change but API.

20face has its face service APIs running on the cloud. However, once the SDK gets to a stable release face service will be replaced by the SDK. That is, the SDK will be running in the cloud sitting between a load balancer and a database. The SDK will leverage micro service architecture when it comes to cloud services. Using the same SDK structure for both the local and the cloud solution has pros and cons. The SDK will have its own profiling, modeling and analytics.

3.2.1 Face Service

20face initially started with face service. It did have almost all the functionality as described under the SDK. The reason to move from Face Service to SDK was mainly the performance issues. Also since 20face is a B2B company other partners were willing to have a local solution as well. Face Service was serving all the customers by then. There were two main goals in building the SDK. First was that to mimic Face Service and the second was safely replacing Face Service with the SDK. While it was an advantage to have a model like Face Service that made SDK development faster, it was a difficult task to replace existing Face Service with the SDK. Not only technical matters should be handled but also Business to Business communication.

3.3 DATABASE IMPLEMENTATION

After that first SDK development meeting I was asked to start working on SDK’s database. Tech Lead Eng. Wout Elferink was guiding the database implementation. Figure 3.1 shows the connectivity of the database to the SDK.

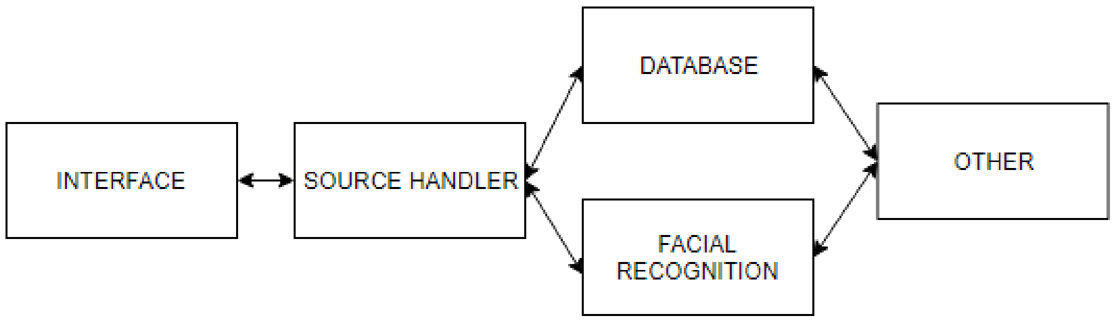

Figure 3.1 SDK composition

3.3.1 Design Review

The design review constitutes the initial and modifications done over the time. The final design review has taken into consideration the following facts.

- Consistency between database file and the memory arrays for read/write data
- Concurrent access to the database file
- Synchronization within local database files after every change in the database
- Performance of the database
- Nearest neighbor search implementations

3.3.2 developing database

For this task I started working on a small pilot project. In that project I built a database using the SQLITE wrapper for C++. The choice of the database manager was made by the Technical Lead. As SQLITE was a lightweight portable single file database manager it was the go to database solution. The reason for implementing it in a pilot project instead of the SDK itself was that the freedom to test and debug. After the pilot project came to a satisfying point it was just a matter of merging it into the SDK with minor changes.

In the pilot project I implemented all the basic read/write functionality. This took about three weeks. Then I started merging the database into the SDK. As for now other functionality that connect with database were not implemented (i.e. Source Handler functionality) Hence, I had to mock those functionality. This was the only time consuming step as integrating pilot project to the SDK was straightforward.

Then it was time to test the database. Testing framework used was Catch2. For setting up the build configuration CMake was used. It was my first time getting hands on these tools. The fundamentals that I have learned in the university helped a lot to setup the test cases to compile via CMake. In particular Maven build setup that I have learned under Web applications and Software Development course helped a lot to understand CMake.

Among other debugs, debugging updating face vector was of major concern. Since it had a lot of read/write functions and insertion/deletion from the memory arrays the initial implementation underwent several modifications. Synchronization of the multiple databases was also taken into consideration. Along with the Technical Lead first we sketched down the possible scenarios. Afterwards solutions were discussed and eventually implemented them. Figure 3.2 addresses most of the scenarios.

All these implementations were done according to the general practices in 20face. I had to adjust coding to the 20face’s coding style. Code style assures readability and flexibility. This made it easier when asking for help, code review and switch tasks within the teammates. Catch2 framework was a lightweight testing framework with its own code style. Later on code style was a blend of 20face and Catch2.

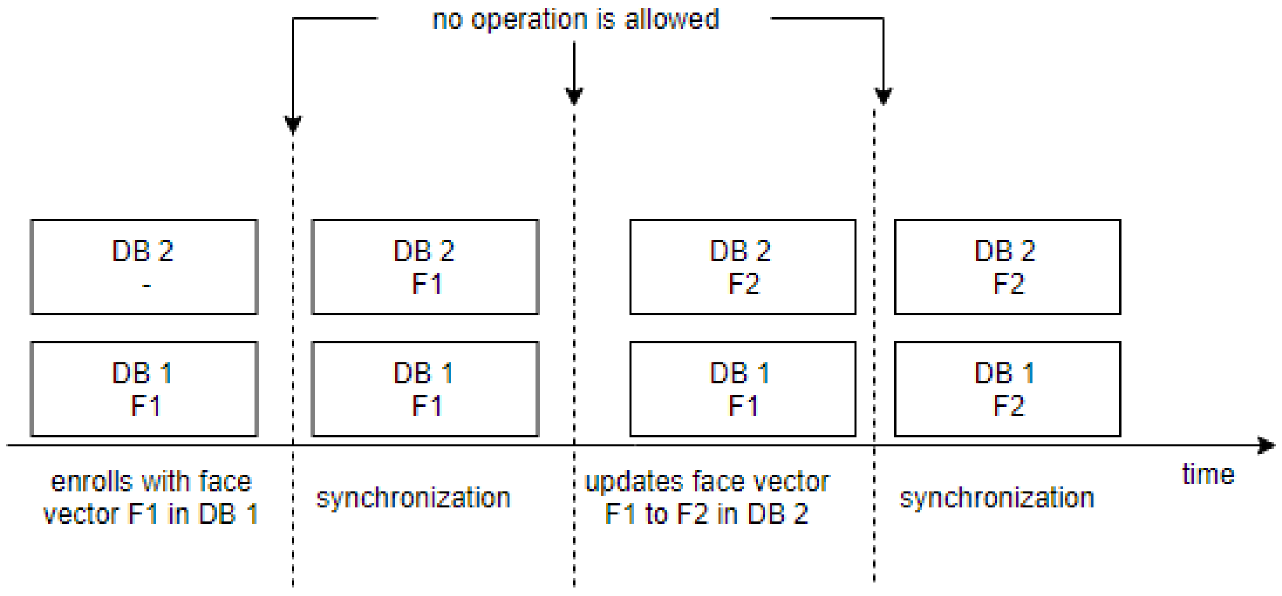

Figure 3.2 Database Synchronization

The version control method used was Bitbucket. The reason behind not to use GitHub was that users have to pay to maintain private repositories. Bitbucket was payment free. Bitbucket and GitHub are more or less similar version control tools. One difference that one could find was that the difference in readMe markup language. Bitbucket had issues in linking in text lines. Also issues related to large file system were met along the way.

There were no quality assurance tools to automatically detect the quality of the code before making a pull request. Therefore, initially I had to update the pull requests due to the failure to maintain code quality. Over the time I could fully adjust to the quality assurance policies.

Jenkins was used for continuous integration. It was under construction along with the SDK. Hence no builds were triggered. I suggested using Travis for the time being. Travis is a lightweight easy to configure setup compared to Jenkins. However, Travis was not free for commercial purposes. What developers did when a pull request was made was that to check out and try to build the branch. This was not the best practice. For an example in my laptop I had header files for LibSodium library for all the different architectures and the others did not. A pull request was failing in other machines but mine. It was after a while the issue was resolved. The original issue came from a CMake configuration.

It was every developer’s duty to review the code and locally build when a pull request is made. C.A Boeren Dev-Ops Engineer ran Jenkins builds after setting up the Jenkins server. Merging pull requests were done after everyone approved it. Since everyone strictly followed the procedure there were minimal rollbacks.

3.3.3 Nearest Neighbor Problem

Finding the closest face vector was a time consuming task. It created bottle-neck in the database pipeline. In order to solve this issue it was proposed to create an index using face vectors. FLANN (Fast Library to Approximate Nearest Neighbor) was used to solve this issue. The downside of this was every time a face vector is added or deleted or updated index tree had to be recreated.

Also initially the database implementation was without memory arrays (i.e. only the database file). After utilizing RAM to store face vectors and related data structures the speed increased. At the same time SDK memory utilization increased. Also this introduced coherent issues.

3.3.4 Implementing Cache

After finishing basic implementation for database and testing it, it was time to optimize the code. One such novel integration was to include a face vector cache. Cache contained eight (can be changed) face vectors. LRU (least recently used) cache policy was implemented in the cache. The idea behind having a cache was to reduce/eliminate full database matches. As there was a high chance that the same person who is in front of the camera can appear in the next frame as well, the cache reduced the search time drastically.

For implementation of the cache bit manipulation techniques were used. The full algorithm was implemented using bit operations. Pablo Garcia - an Intern helped with the implementation.

3.3.5 Measuring Performance

Chief Technical Officer asked me to conduct a performance test for the overall SDK and only for the database part separately. It was a continuous process. For each new version of the SDK the performance tests should be run, evaluated and documented. For documentation Confluence Pages was used. It was my first time creating official documents. I had to update the SDK user guide and a separate document for performance. Report writing skills that I gained in the bachelor’s course were put into action.

3.4 WRITING ANDROID WRAPPER

The next main task that was assigned to me after database implementation was writing a wrapper for Android. As Compo - the main business customer of 20face has requested that the SDK should also run on mobiles, Technical Lead set that task to me. I had experience with Java but Android. The work I had to do was not pure Android. It was in between native (C/C++) and Java which is call JNI(Java Native Interface) . Technical Lead laid foundation by creating a sample Android application. He also implemented a couple of JNI functions as a start. This made it easy for me to follow and implement the rest of the JNI functions.

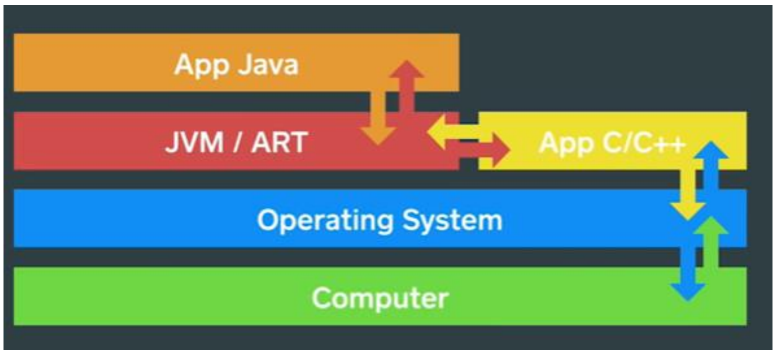

Figure 3.3 JNI bridge between JVM and Native app

The first thing I had to do was to learn JNI. It was hard as its documentation [3] said. It took me about one week to get familiar with basic JNI. On the other hand, I got the opportunity to learn a new low level language. As shown in the figure 3.3 JNI acts as the interface between JVM and native application. There are facts to be considered when writing JNI. To get the maximum speed from native code minimal amount of JNI calls should be made. Technical Lead helped a lot to get JNI code up and running. Once we had a problem with catching exceptions. After inspection it was found out that JVM lagged to throw the exception (i.e. even after an exception is raised in the native code JNI tends to proceed ending up being crashed). This scenario is illustrated in the figure 3.4. It was assigned to me as a responsibility to keep the JNI code up to date with the SDK.

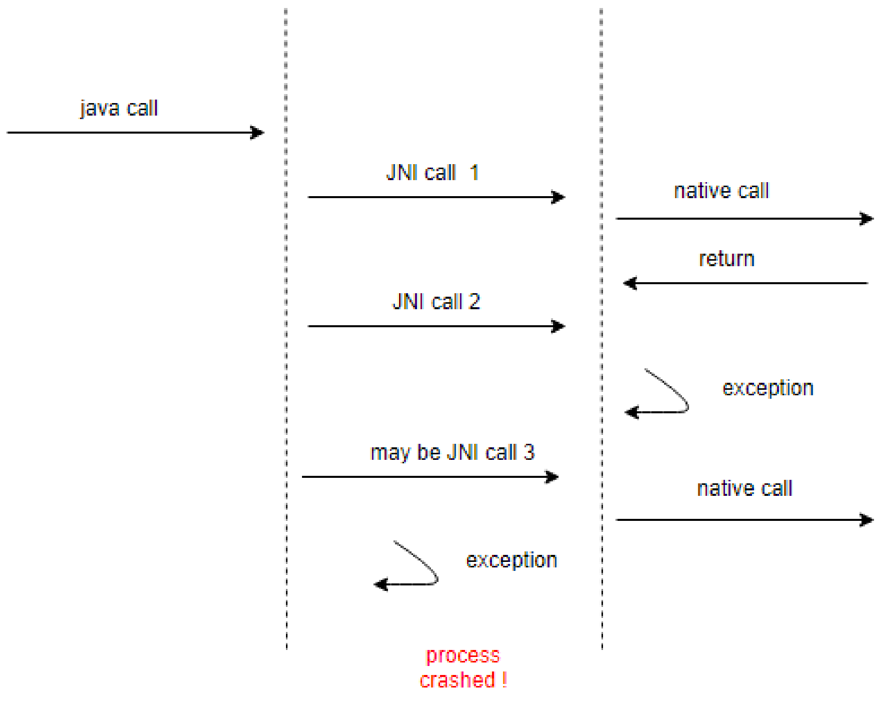

Figure 3.4 JNI keeps calling native functions not aware of the exception

3.5 RECONFIGURATION OF CMAKE

CMake is the building tool used in the SDK. It was getting complicated (figure 3.5) and bulky over time due to the addition of different components. The components were python wrapper, android wrapper and test cases. Android wrapper had to be built for different architectures. This made CMake file even more complex. I proposed to work on reconfiguring CMake file. Technical Lead accepted it and assigned me to work on it. I followed an approach similar to librealsense SDK [4].

Soon after reconfiguring CMake it was much easier to add new components. These new components were License Generator, Model Encryptor, Android example application and Java wrapper.

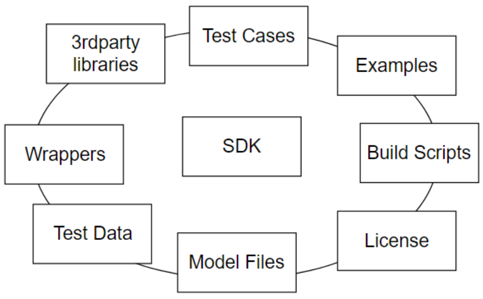

Figure 3.5 SDK and its components

3.5.1 Building License Generator

Initially license generation was an inbuilt tool of the SDK and only accessible in debug version. Instead of that it was proposed to take out the functionality and use CMake to configure whether to compile license generator code. In that way license generator resided inside the SDK code base but not a part of the SDK. It was built as a separate executable with a couple of test cases.

3.5.2 Building Model Encryptor

Same as in the case of license generator initially encryption of model was inside the SDK. It was accessible only in debug mode. To get the maximum benefit of the CMake it was proposed to take out model encryptor out of the SDK as well. Moreover, the decryption key (SDK used AES encryption) was passed as a CMake variable.

3.5.3 Android Example Application

With the idea of testing the Android wrapper in Jenkins, Dev-Ops Engineer proposed to build an Android application with the SDK build. As the application was already maintained in a separate repository it was a matter of configuring CMake and Gradle builds. Then the Android emulator had to be installed to Jenkins. As for then there were no test cases specific for Android. Espresso testing framework was used to create some test cases for the application. Following figure shows how Android application was maintained and updated across three repositories.

3.5.4 Writing Java Wrapper

By then the SDK supported C++, Python and Android. Technical Lead assigned the task of writing a Java wrapper for the SDK. Since the JNI code was more or less similar to what was in the Android wrapper, writing the Java wrapper was straightforward. I also implemented test cases to test the JNI code. The test cases were written in Java. Adapter software design pattern (figure 3.6) was followed in creating the Java wrapper. Under the course - Software Engineering I had learned software design patterns.

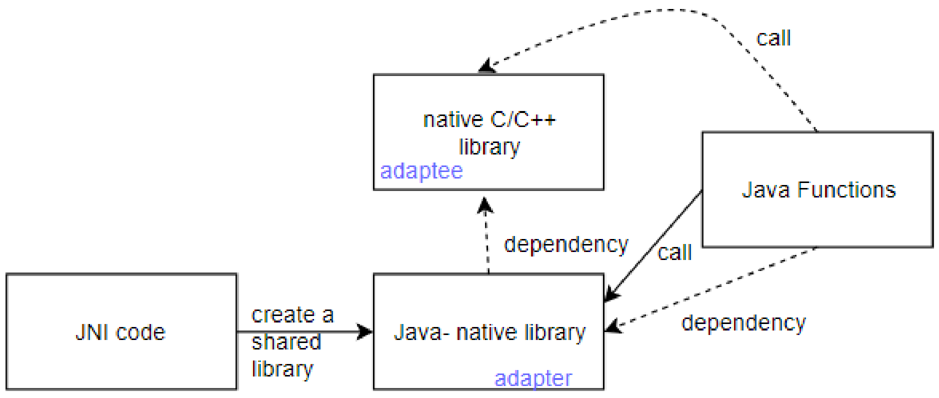

Figure 3.6 Adapter Software Pattern in Java Wrapper

3.6 EXPERIENCE GAINED

For the first two months I was in a continuous process of learning new software and best practices. I learned CMake to a point where I was given the task of reconfiguring the main CMake file to ease future builds. I really enjoyed handling memory pointers and debugging JNI code, which again a new thing that I learned. Also I could put what I have studied in the university to action. By doing that I learned how to handle practical scenarios. To elaborate on that it is not the same in what we learned in theory when it came to practice.

Another important thing that I learned was planning and designing before implementation. The SDK went through multiple review cycles before it was put into development. I learned the value of project planning and project meeting and that they are equally or more important than the actual implementation. I learned how to be precise on my work and be brief on describing the work. This effectively reduced the time a new developer has to spend understanding the existing code.

Learning JNI and using native code to run on Android applications efficiently and effectively helped me a lot to initiate my final year project. Also on the last day I got the chance to do a small presentation on Java and Android wrappers to the developer who is going to take over my work. I developed the skills to how to effectively transfer knowledge. Writing documentation for the users and the developers developed my writing skills. This will help me write my project reports in the future.

In particular the prior knowledge I had about object oriented programming, software patterns, networking, memory handling served me understanding the context better. Above all, I value the communication skills that were developed during the time of the development of the SDK. I learned how to professionally ask for help, excuse and point out errors. Eventually I gained the experience to how to collaborate with people with different skills and mindsets.

Chapter 4

MINI PROJECT – GETTING STARTED WITH KUBEFLOW

4.1 INTRODUCTION

After implementing Java wrapper my work on the SDK was almost complete. Except for minor updated and debugs I had time to learn and assist other on going projects. Chief Technical Officer gave me the chance to select such an ongoing project. Since I was excited to learn more about continuous integration I wanted work along with the Dev-Ops team. At that time Dev-Ops team was busy with helping research team to build a pipeline for training the 20face face model. As for then the models were trained in the university resources. It was time consuming and the research team had to wait for the results before doing improvements.

4.2 WHAT IS KUBEFLOW

Kubeflow was created by Google. Then later on it was made open-source. It is a hot topic these days. Basically Kubeflow is a machine learning tool. It helps to create machine learning pipelines. The pipeline constitutes of components. Component can be thought of as a docker container with specific inputs and outputs. One main goal of using Kubeflow for machine learning applications is that it supports direct deployment into Kubernetes - a container orchestration tool. Refer figure 4.1 to get a brief idea about Kubeflow and other related cloud services.

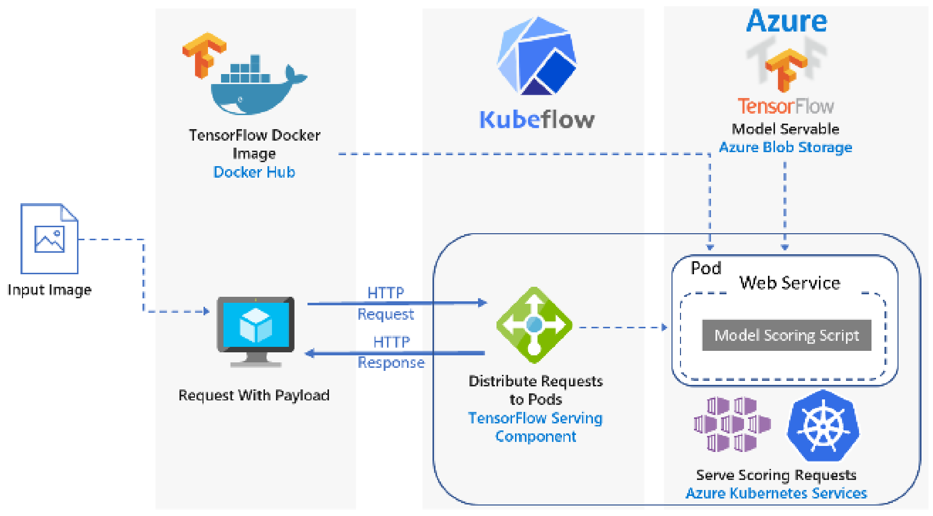

Figure 4.1 Kubeflow fits in a group of cloud services

4.3 GETTING STARTED WITH MINIKF

I had the chance to spend some time getting to know the technology around Kubeflow. Since it involves cloud resources and cloud services cost money, it was decided to get it up and running at the lowest cost possible. C.A Boeren - Dev-Ops Engineer suggested that instead of directly approaching Kubeflow it is better to investigate it locally. Since everyone were new to this field no specific objectives were set. The goal was to be comfortable with Kubeflow as much as possible.

4.3.1 miniKF

In order to use Kubeflow the first thing was to understand the nature of it. Since no one in the team really excelled in it the best approacch was to spend some time playing around with it. The drawback with Kubeflow was that it was an online tool. To even run a test pipeline the components were needed to be hosted in a cloud. Hence MiniKF [5] was used. MiniKF is a local solution for testing. It replicates Kubernetes server with miniKube. For data storage it uses Rock data manager. With these two combinations it builds a mini version of kubeflow, which is called MiniKF.

4.3.2 Creating a Tutorial on miniKF

Since Kubeflow was introduced in 2018 and MiniKF was developed very recently there were hardly any tutorials to get a simple pipeline up and running in MiniKF. After a lot of search I decided to do a pilot project to create a simple pipeline and eventually push it into GitHub (Annexure 1). I made the readMe file more like a tutorial. In the project I created a pipleline (figure 4.2) with three components and two of them are identical. The components were hosted in DockerHub.

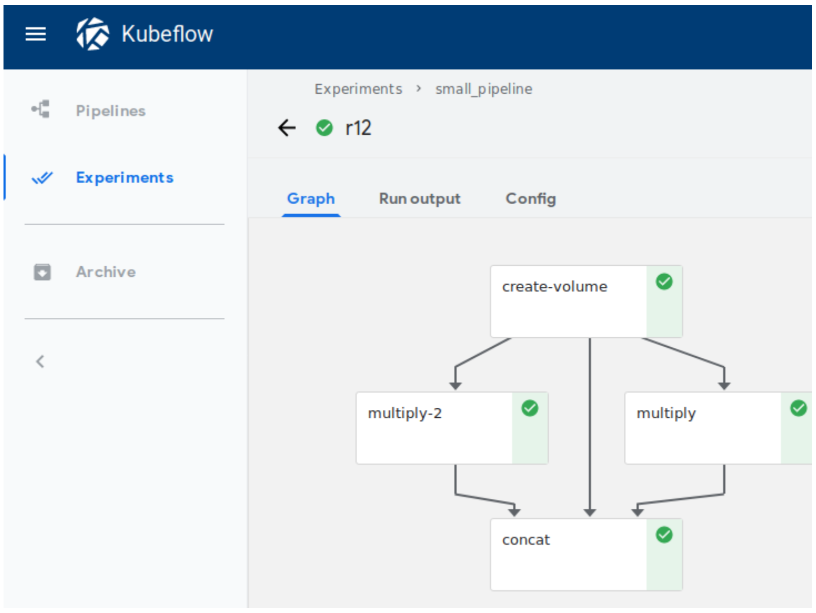

Figure 4.2 Example Pipeline created using MiniKF

I have described the way to get the maximum benefit from miniKF without hosting components in Google cloud storage or any such environment. The limitation imposed in miniKF was that it had Rock dependency. In real production environment building a pipeline was way easier than in miniKF. Therefore it was decided to use miniKube straight away. Doing that the principles of Kubeflow could still be tested locally but with more control.

4.4 FUTURE WORK

It was in my last days that I started working on this project. Hence, I could not wait until the project is over. The main objective was to setup a training pipeline on AWS and use it for training the models. 20face was making good progress towards it.

4.5 EXPERIENCE GAINED

This mini project gave me the opportunity to learn cloud based tools such as Kubernetes, AWS and Kubeflow. Also I could improve my knowledge on the production environment of data science field. I could write a tutorial on how to build a pipeline using miniKF. This gave me the chance to collaborate with developers that I have not engaged with before. Also I could observe the fact that Dev-Ops team was more careful with their development than the development team, may be because they were much closer to the operational level of the business.

CONCLUSION

Internship was the first hands on experience after studying in the university for three years. It was a great opportunity to try out and apply the knowledge gained in the university. In particular I could learn what an SDK is, writing JNI, optimization and much more during my first project - building the 20faceSDK. In the second project I learned Docker, Kubernetes, Kubeflow and continuous integration concepts. I was not even aware of the existence of most of these technologies and concepts before. My internship was a continuous learning process. I had to learn many things before applying. If the time duration is divided into halves, one half was spent merely learning new things. However, the fundamentals that I have studied in my bachelors laid the foundation to learn new technology faster.

Adjusting the working environment and implementing solutions to real world problems were new experiences to me. I got to know the current trends in the industry. I can now build my career with more informed decisions. Also working with different people and maintaining professional relationships with them was a new experience too. Writing emails, sending messages in workspaces like SLACK and professionally asking for help were a few areas that I practiced during my internship.

I learned to work according to a plan. Sprint plans we had in the SDK development was a main motivation for it. Also I learned how to effectively and efficiently work in the week days and to rest on the weekends. Not thinking about the work except for the working hours made me got to work fresh and energetic. It also helped to get the maximum out of the working hours. I learned to work with more responsibility and to be more concerned about the deadlines.

I learned a lot of best software practices from my colleague C.A Boeren - Dev-Ops engineer and Pablo Garcia - an Intern. We helped each other and the learning never stopped. Also Wout Elferink - Technical Lead helped me when I had difficulties with Android wrapper. In overall everyone in the office was very supportive and encouraging.

The social environment around the office helped me to love what I was doing. The habits of eating lunch together, celebrating birthday parties etc. boosted up the affiliation to the company. I presume the content I had for my internship was sufficient in qualitative and quantitative wise. It was a shame to leave the office after working there for six long months and more importantly I enjoyed every bit of it.

REFERENCES

[1] “20FACE” – 20FACE, [Online]
https://www.20face.com

[2] “20FACE ECOSYSTEM” – 20FACE, [Online]
https://www.20face.com/#our_vision

[3] “JNI“– NTU, [Online]
https:// https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaNativeInterface.html

[4] “librealsense” – GitHub, [Online]
https:// https://github.com/IntelRealSense/librealsense

[5] “MiniKF” – kubeflow, [Online]
https:// https://www.kubeflow.org/docs/other-guides/virtual-dev/getting-started-minikf/

ANNEXURE

Annexure 1

Tutorial based on getting started with a simple example pipeline using MiniKF

Repository: https://github.com/hiruna72/miniKF_example_pipeline