BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS 399

SUMMER TRAINING REPORT

Muhammad Arham Khan

21701848

Performed at

Cubicl

June 15, 2020 – September 15, 2020

Table of Contents

1 Introduction

This is a detailed report of my summer internship as a Software Engineering Intern at Cubicl[1] (officially registered as Z Yazilim in ODTU Teknokent, Ankara) from 15th June 2020 to 15th September 2020.

The company owns and develops Cubicl, a cloud-based task and client interaction management system for the general public and specialized variants for businesses. The company also offers contract-based development services for bespoke application development.

My motivation behind choosing Cubicl was the scope of working on a mission ciritical product that is used by over 500 firms on a regular basis to manage project tasks, employee time schedules and regular communications. Also, I saw an opportunity to learn TypeScript and improve my hybrid mobile application development skills by developing a product from scratch. Hence, realizing this to be a profitable opportunity from a learning standpoint, I was inclined towards interning at Cubicl.

As a part of the Mobile Development team, I was responsible to learn TypeScript and React Hooks (I was already experienced in React Native[2] so didn’t have to learn that) and then utilize it to develop the latest version of Cubicl’s mobile application. The current application was developed using NativeScript and was outdated according to current market standards and since the company had limited time to deploy the new application, it was decided to proceed with hybrid application development using react native.

The project I worked was significant to the company and in general because in today’s fast-moving world, clients utilizing the Cubicl[1] platform for their task management need a robust and easy-to-use mobile application. Without such an app, users usually had to rely on the web app which although worked perfectly, didn’t provide the user the same level of native operation as an application would. Hence, to deliver a better service to the clients and have a more robust and efficient product, Cubicl decided to develop the application from scratch and I got the opportunity to be a part of the development team.

2 Company Information

2.1 About the company

Cubicl[1] is a software development firm (officially registered as Z Yazilim) that’s core product is a cloud-based task and client interaction management platform. The company is based out of Ankara with their office located in ODTU Teknokent. The company also provides contract-based bespoke application development opportunities for the

The Cubicl[1] platform is available as a web application at (htttps://cubicl.io) and mobile applications are available Android and iOS app stores. Many firms have been using Cubicl on a regular basis to manage their day-to-day operations. Some of the prominent customers of Cubicl include:

- Vestel
- Cepmoda
- Beykent University
- Creasaur

As of recent, the company has also started integrating wrappers to allow integrated working with widely-used platforms like Google Drive, WhatsApp, Slack, Github etc.

During the Corona outbreak, all of the development was done remotely and the company utilized their own platform (Cubicl) to manage the tasks and remote management of the employees and projects.

2.2 About your department

I was part of the Mobile Development Team in the Engineering and Development department. The engineering and development department was headed by Mr. Erkan Arslan (the company's owner) and he supervised our development pace and major project decisions. The core duties of the engineering department were developing and maintaining the Cubicl platform and also handle other contract-based development for clients.

The Mobile Development team was led by Mr. Çaglar Keskin, where his core duties involved monitoring feature development and taking code strategies decision to ensure a smooth development of the product. He was also responsible to merge and manage the code repositories and control the branches. The core responsibilities of the Mobile development team were to develop and maintain the Cubicl platform's mobile application and also handle other contract-based development for clients.

Since I was interested in Full-stack development, although I joined as a mobile application developer, I was encouraged by my supervisor to handle some backend development tasks (on the Laravel 5.0[3] based backend system) needed for new features in the mobile application.

2.3 About the hardware and software systems

As a mobile application develop, I was provided with a Macbook Pro having the following specifications:

- Intel Core i5 7th generation processor
- 8 GB ram
- 256GB SSD and 1TB external hard drive

The software suites we were provided for development were as follows:

- XCode
- Android Studio
- Visual Studio Code, which we utilized to develop and test the React-native application
- Git Version Control System

2.4 About your supervisor

I had two supervisors guiding me through my internship program. Their details are:

1. Mr. Erkan Arslan:
    - Position: CEO and Engineering Team Lead
    - Graduated from: Middle Eastern Technical University, 2017
    - B.S. in Computer Engineering
    - M.S in Electrical Engineering

2. Mr. Çaglar Keskin:
    - Position Mobile App Development Team Lead
    - Graduated from: Kirikkale University, 2016
    - B.S. Computer Engineering

3 Work done

3.1 Learning new technologies

Considering that I was experienced in JavaScript and hybrid mobile application development using React native, I was in a good standing to start the development process right away and learn TypeScript[4] and React Hooks[5] on the way. But to ensure a smooth process, my supervisor suggested I take one week to acquaint myself to TypeScript and React Hooks. I spent this time using tutorials from Toptal Developer blogs and official React Native documentation and developing a sample login workflow using both technologies.

During the week, I learnt the differences between Javascript and Typescript[4] and how Typescript was a more robust alternative to it. I also realized the importance of function based components (the approach is called using React hooks) to make the application development easier and code more readable. Finally, I got a chance to acquaint myself with babel parsers for TypeScript[4] and how object types are differentiated among different data structures in TypeScript.

3.2 Phase 1 – Mobile Application Development

3.2.1 Component Design

As we started developing the project, the first aim was to develop static components using the new app design. We had a design layout available at Zeplin and we were assigned regular tasks to create one scree using static JSX Elements to finish the design phase of the application.

During this phase, we had a basic static layout of the React Native application with basic packages like react navigation and core react-native installed and the default route was edited to the current screen in development to test. We mostly used VSCode with TypeScript babel enabled to develop the JSX Components with static text elements. To validate the code and design of the components, we used to assign control tasks to other team members and had a design control phase, which was controlled by the designer herself. This way we knew that the app components being developed were cohesive and complied with the original design provided.

To ensure that the designed components looked the same on Android and iOS devices, we had to test the components on both devices before sending it for control. Since React-native[2] focuses on native implementation for the components, I had to write multiple platform-specific component check to ensure it looked the same across the different platforms.

To provide a native and uniform user interface across the application, we focused on developing common components with valid props that could be used in multiples screens. For instance, we used the react-native-calendars[7] package base to create our own custom calendar component that could be used with the visible dates prop and monthChanged callback to display tasks calendar, personal tasks calendar, and employee attendance calendar chart. Similarly, we utilized standardized packages like react-native-modal, action-sheet etc to ensure that although multiple developers were working on different parts of the application, the core feel and usability of the product remained the same.

Since the components were individually designed, it was impractical to design each and every component of the application without being able to test its workflow and its integration with the service files. Hence, we developed the Main Screens and layouts and skipped minute elements like popups, loaders, animations etc. at this stage.

3.2.2 Core logic refactoring

Since we already had an existing web application developed in AngularJS (for the front-end), we had most of the core logic available for the backend update and access services written in javascript. But to integrate that service files, we needed to refactor the Javascript code to include react native libraries and convert it to TypeScript to be parsed by TS babel. Hence in this phase of development, our team of three engineers refactored service files according to TS and ReactNative[2] standards. Although the core logic was the same, we had to make quite a lot of changes to the service files. For instance, the previous services required instantiated use of HttpClient library from AngularJS whereas in React Native we planned on using Axios[8] request management package to ensure persistent and robust request queues. Furthermore, since the application followed a Publisher-Subscriber model to update components in the app and receive data callbacks, we had to include RxJS[9] in the React Native service files to make it functional.

Another major change that was necessary was to counter duplicate initialization of the service files on component re-render in the application, we had to make the service file initialization in the application persistent and also ensure that dormant service files were destroyed to reduce memory usage. Hence, we utilized queue structures to ensure that no more than 12 services were initialized at any moment and that if there were any existing services initialized already, it was used instead of reinitializing it. At an early stage in the development phase, we had decided not to use redux[10] and redux-saga middleware to ensure code cohesion with that of the web application. So, instead we decided to use service objects and their properties to maintain app-wide states and pub-sub model to relay information across the application. This way we were able to follow the same strategy we used in the application and kept the run-time impact of the application low as well since there were no app-wide app data stores being maintained.

3.2.3 Application Development and Navigation Integration

Once the core service logic files were ready, we started working on integration the logic elements in the static design components and making them dynamic. Since the application was a complicated one, each develop had to work on a variety of components since the first day and had to see it through. For instance, since I developed many static components like calendar component, tasks screen, clients screen, chat screen, chat threads screen, profiles screen, personal notes screen etc., I also had to refactor the relevant service files for it and then integrate the core logic into the static components for these screens.

During this phase I developed the core run-time logic for component life cycle events and user interaction events including component initialized, component destroyed, user interactions, data filters and search, app focus changed, navigation events etc. Since the applications utilized React Hooks[5] instead of conventional component lifecycle events, it was a challenge maintaining the app state using variables and I had to use useRef and useState constants to get latest variables and trigger component updates. Furthermore, since there were asynchronous data access operations using axios[8] in the server files, I had to include multiple useEffect functions to update the component layout on data retrieval.

Finally, once the components were developed, we had to test their usage on both Android and iOS devices in multiple lifecycle cases like app in background, screen navigated back etc. We also had to test the core update logics in test cases that had already been observed in the web application by previous engineers. Hence in our regular scrum meetings, we used to discuss our current components and verify them against previous observed and fixed bugs in the web application to ensure that the same bugs did not appear in the new product.

Since we used to work on separate branches for each screen, I also got to experience using Git features like rebase and merge requests on separate branches to integrate other branches in the current branch and then work on navigation logic. Since it was a complex application, we had to use multiple navigators to ensure the expected application workflow. For instance, we used TabBarNavigator for the main dashboard screen, SwitchNavigators for the login screen logic, DrawerNavigator for the main drawer menu and finally StackNavigator for the futher navigated screens like task details screen.

3.2.4 Backend Development

Since I was interested in Full-stack development since the first day and had experience working with Laravel and backend service development, when we had to integrate OneSignal SDK[11] in the backend application and add feature logic for personal tasks screen (a new feature), I was asked by my supervisor to handle these tasks. Hence, I utilized an opensource integration wrapper called Laravel-onesignal[3] to handle sending notifications to the relevant user. Now since the backend was already storing user's playerID everytime they logged in, I had to access the user, see if the user was online on any of the devices and if so, send a notification dispatch to that device only. Otherwise, I dispatched a notification to all devices registered with the requested user. To implement this, I had to implement a new controller called OSNotificationsController to dispatch notifications to the mobile devices. To enable off-thread dispatch, I also had to create a notification dispatch service in the Laravel application and a Queue Job for it.

Apart from the backend development needed for notification dispatch, I also added the code (models, controllers etc.) and the relevant database migrations needed for the personal tasks screen. The controller create the basic CRUD functions for the personal tasks but also included sort by dates, search by name tag, bulk complete and bulk export functions for the tasks.

3.2.5 Real-time updates

Once the application had been developed, there was a core necessity to integrate real-time updates in the application to enable features like chatting, task status update, task activities etc. Since we decided on using OneSignal SDK[11] to handle the notifications, I was responsible for integration and linking the OneSignal SDK with iOS and Android builds and writing notification handling logic to route to the relevant RxJS[9] subject. This was, services subscribed to the NewNotification Subject would receive the notification relevant to that service. Apart from this, I also had to develop the background notification handling logic to route new data to the specified activity.

3.3 Phase 2 – Testing and Deployment

After completing the application in the first 10 weeks, we took the next two weeks to fix the bugs in the app and deploy it.

3.3.1 Bug fixing

We released application's an alpha version (apk file for android and aab bundle on TestFlight for iOS users) so that company employees could use, test and report any bugs with the application. Naturally, many new bugs were reported and we utilized the Cubicl platform to manage the reported bugs and assign them to members of the team. Some bugs were trivial design edits like content not fitting a smaller screen or text data overflow. Whereas some bugs involved the app crashing in edge use cases. But considering that the app was developed from scratch and the person fixing the bugs was familiar with the possible causes for the bug, we were able to rectify most of the available bugs within the two week timespan.

3.3.2 Native bundle compilation

Once the react native app was ready and bug free, it was my responsibility to develop the android bundle from the JSbundle code and also assist for the iOS bundle. We added the npm packages to linked libraries for both android and iOS project outputs and then added the relevant keychain access and keystore accesses to the environment. Hence, after every thing up and running, we just added the key SSH keys to the setup files in the application and produced the native bundle archives that could be deployed to the app stores.

3.3.3 Deployment

Once bundles were finalized, we deployed the packages with new details about the re-designed application and the relevant screenshots. It took around 3 days for both, the android and iOS, applications to go live on the store but they were a successful product produced and deployed in the promised time.

4 Performance and Outcomes

4.1 Solving Complex Engineering Problems

Since the Cubicl platform comprised of complex features like task assignment, schedule planning, real-time component updates, messaging, stacked file systems, and multi-client and multi-device access, we had to face a wide variety of engineering problems during the development process. For instance, early on during the refactoring phase of core service logic, we realized that since react-native components re-rendered on state updates, we needed a way to block repetitive service initialization. To reduce memory occupied by the application, had to maintain a fixed length queue of initialized services and kill the older ones. Another engineering problem that we solved was receiving real-time updates on multiple active devices at the same time and parsing it as a background notification on dormant devices, so we maintained a HashSet of active devices every time the device opened up and if the device was active, the user appeared online and background update notifications were sent to the user, otherwise background pop-up notifications are sent to the user. So considering the complexity of the application, there were a lot of engineering problems that we had to face on a day-to-day basis.

4.2 Ethical and Professional Responsibilities

During the course of my internship, I was introduced to some professional and ethical issues that only working in a corporate company with a major user-base could get you. To begin with, the issue of user data privacy was significantly highlighted throughout the internship and was a major player in a majority of the team decisions about projects. Furthermore, there was the case of code privacy where a holistic NDA and team training every now and then and packaging solutions ensured that the proprietary code being developed at the company remained within the walls. Furthermore, I also learned about my responsibility as an engineer to ensure that the deployed product and user data was not compromised during my development. I also learned to ensure utilizing development server environments to test an upcoming product while using the latest information. To do this, I had to use a cron job and push notification context on the dev server that received notifications from the original server and pushed changes to the development server so the data was always current but did not interfere with the data on the main server. Hence, as a professional engineer, I learned to ensure data was secure and unaltered by my development. Finally, I also learned to write clean, maintainable code and following a style guide enforced by the company. This way, I was fulfilling my responsibility to the company and the other engineers that would follow me.

4.3 Making informed judgements

During the course of my internship, although the product I was working on didn’t have a global impact, I had to utilize my knowledge of the global, economic and societal contexts to modify my product. For instance, realizing a the necessity for the product I was working on globally for task management and seeing a future in it, we chose to add localized-strings in the application so that apart from the current English and Turkish translations of the application, any other language could be added in the future. Furthermore, considering the fast-moving economy of the world these day and the increasing reliance of firms on technology to manage their day-to-day affairs, and the amount of impact that any discrepancy in a platform like Cubicl[1] could have on a user’s workflow. Hence, throughout the development process, it was ensured that the application was as efficient as possible and didn’t rely on high-end devices to run well. Considering that the application would be used by firms of all size, by people of all background to work towards a unified goal of automating their task management, we had to ensure a cross-platform availability and low-device-specification requirement.

4.4 Acquiring New Knowledge Using Appropriate Learning Strategies

Since I was only experienced in React Native and Javascript, I had to learn multiple things like TypeScript, React Hooks, OneSignal[11] integration, native library integration in React Native and shared component development. Now since I understand the importance of learning through experience, instead of simply reading through the documentation only, I learned the tools by developing a sample test application using React Hooks and TypeScript. I also relied heavily on React Native and Toptal’s developer blogs to learn good development practices and learn the tools properly. I mostly used FreeCodeCamp to learn TypeScript and its conventions. Since OneSignal didn’t need any prior reading or experience, I learned using it while following the official documentation and tutorials off the internet while working on the completed Cubicl application. Hence, I used a variety of methods to learn the required tools but ultimately, I also learned what approach works best to learn what kind of tools.

During the branch merge and rebase phase, I realized that although the Github CLI was a powerful tool, I still needed to use an external software to better understand and monitor the process. Hence, I chose to learn using GitKracken, a Git management client. With it I was able to manage and monitor the branches better and handle merge and rebase operations in a more graphical and intuitive way and not make any errors.

4.5 Applying New Knowledge When Needed

Since I learned how to use TypeScript and ReactHooks[5], I realized the importance of using types to create a robust codebase and hence, developed the application with explicit type and object path declaration. It is not a convention to use object models in React Native, but since we wanted to develop a robust yet readable codebase, I relied on using Models and typecasting all received request data through those models.

Additionally, I was used to using component update functions to handle lifecycle updates in the application. But in this internship, I learned the value of functional programming and the use of useState, useRef, useEffect and other React Hooks to ensure a readable code and easier app life-cycle workflow. But since it is not always necessary to use all the hooks in conjunction, I learned to use the relevant tools in the necessary places as well.

5 Conclusions

My internship as a Software Engineering Intern at Cubicl was a remarkable way for me to practice the knowledge I acquired at Bilkent University in a professional, real world environment. Not only did I get to learn new technologies and frameworks that are in high demand in the job market, but also got a sense of corporate culture and team-collaboration in a professional environment. So, all in all, I believe that this internship, supplemented with my learnings at Bilkent, would greatly benefit me in my future endeavors and I am happy to have worked for a company with products that impact the mass-user on a daily basis.

References

[1] “Cubicl”
https://cubicl.io/ [Accessed Oct 16, 2020]

[2] “React Native”
https://reactnative.dev/ [Accessed Oct 16, 2020]

[3] “Laravel”
https://laravel.com/ [Accessed Oct 16, 2020]

[4] “TypeScript”
https://www.typescriptlang.org/ [Accessed Oct 16, 2020]

[5] “React Hooks”
https://reactjs.org/docs/hooks-intro.html [Accessed Oct 16, 2020]

[6] “Working with React Hooks and Typescript”
https://www.toptal.com/react/react-hooks-typescript-example/ [Accessed Oct 16, 2020]

[7] “Action Sheet”
https://github.com/beefe/react-native-actionsheet [Accessed Oct 16, 2020]

[8] “Axios”
https://github.com/axios/axios [Accessed Oct 16, 2020]

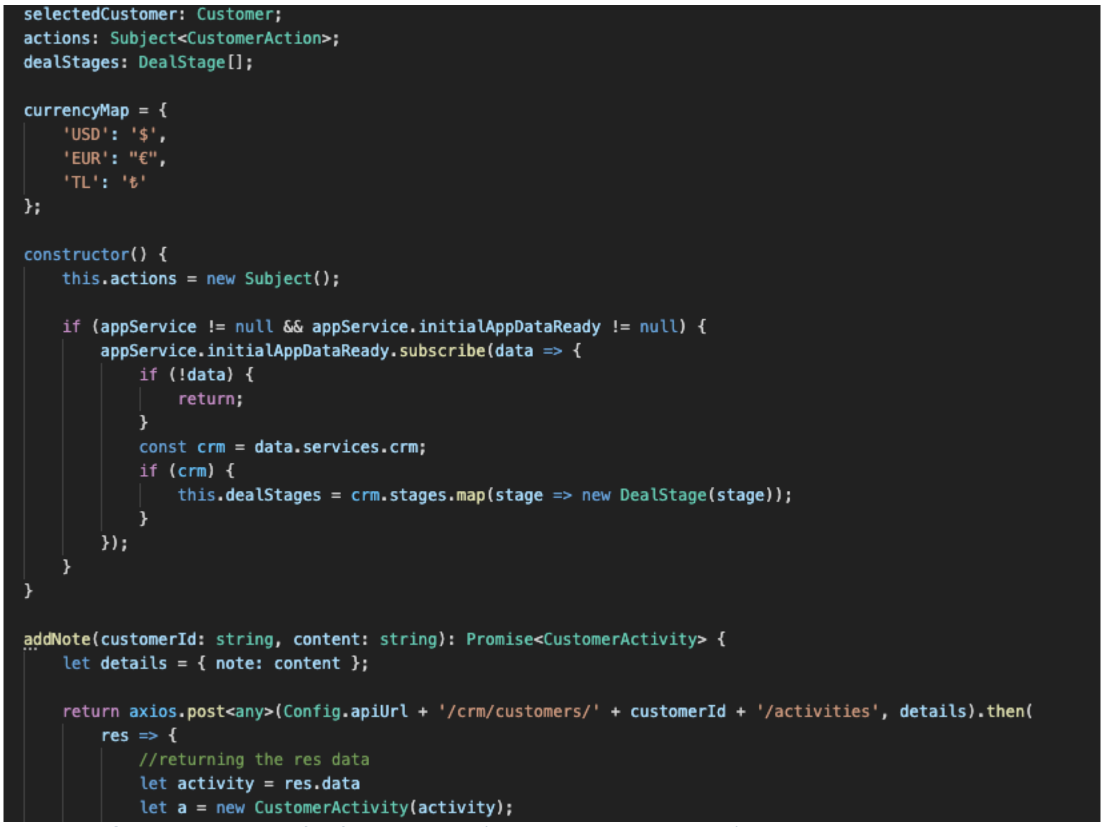
[9] “RXJS”
https://www.learnrxjs.io/ [Accessed Oct 16, 2020]

[10] “Redux”
https://redux.js.org/ [Accessed Oct 16, 2020]

[10] “Onesignal”
https://onesignal.com/ [Accessed Oct 16, 2020]

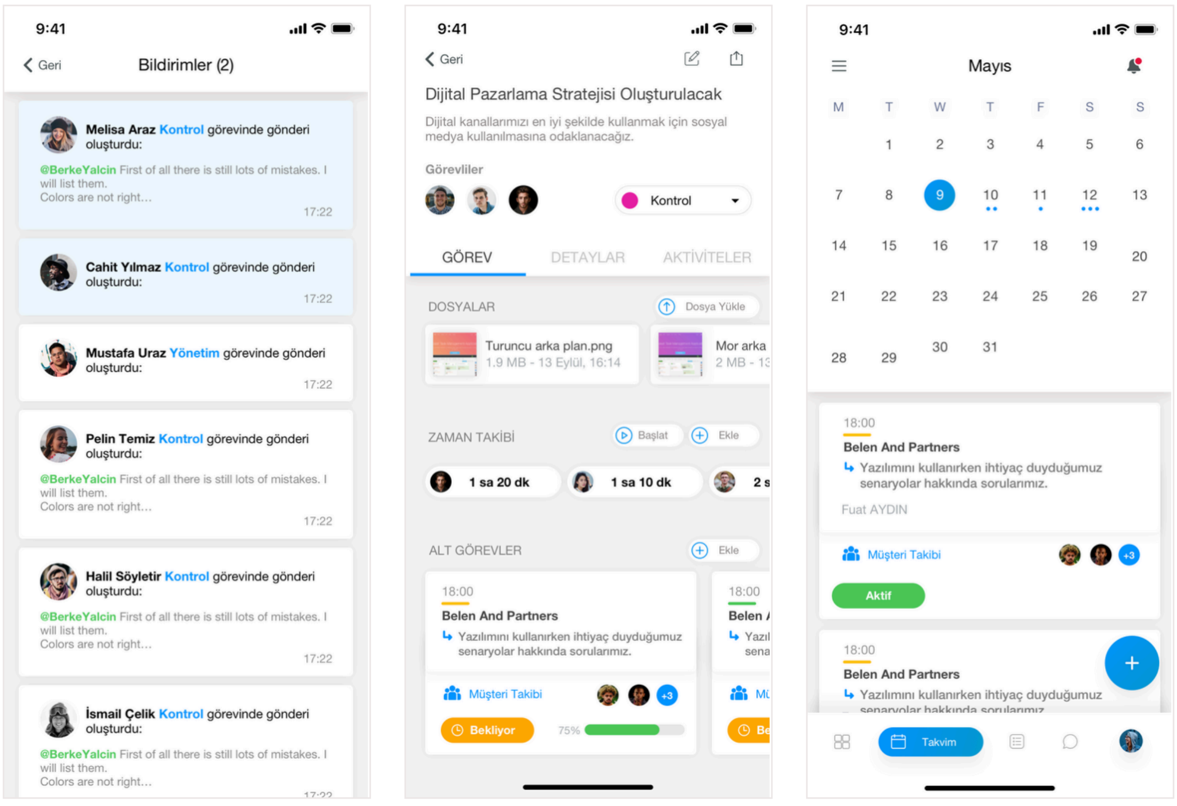
[11] “GitKraken”
https://www.gitkraken.com/ [Accessed Oct 16, 2020]

Appendices



```
selectedCustomer: Customer;
actions: Subject<CustomerAction>;
dealStages: DealStage[];

currencyMap = {
    'USD': '$',
    'EUR': "€",
    'TL': '₺'
};

constructor() {
    this.actions = new Subject();

    if (appService != null && appService.initialAppDataReady != null) {
        appService.initialAppDataReady.subscribe(data => {
            if (!data) {
                return;
            }
            const crm = data.services.crm;
            if (crm) {
                this.dealStages = crm.stages.map(stage => new DealStage(stage));
            }
        });
    }
}

addNote(customerId: string, content: string): Promise<CustomerActivity> {
    let details = { note: content };

    return axios.post<any>(Config.apiUrl + '/crm/customers/' + customerId + '/activities', details).then(
        res => {
            //returning the res data
            let activity = res.data
            let a = new CustomerActivity(activity);
```

Figure 1: Customer service file from the app (using Axios, Typescript)



9:41
Geri
Bildirimler (2)

Melisa Araz Kontrol görevinde gönderi oluşturdu:
@BerkeYalcin First of all there is still lots of mistakes. I will list them.
Colors are not right…
17:22

Cahit Yılmaz Kontrol görevinde gönderi oluşturdu:
17:22

Mustafa Uraz Yönetim görevinde gönderi oluşturdu:
17:22

Pelin Temiz Kontrol görevinde gönderi oluşturdu:
@BerkeYalcin First of all there is still lots of mistakes. I will list them.
Colors are not right…
17:22

Halil Söyletir Kontrol görevinde gönderi oluşturdu:
@BerkeYalcin First of all there is still lots of mistakes. I will list them.
Colors are not right…
17:22

İsmail Çelik Kontrol görevinde gönderi oluşturdu:
@BerkeYalcin First of all there is still lots of mistakes. I will list them.
Colors are not right…

9:41
Geri
Dijital Pazarlama Stratejisi Oluşturulacak
Dijital kanallarımızı en iyi şekilde kullanmak için sosyal medya kullanılmasına odaklanacağız.
Görevliler
Kontrol
GÖREV DETAYLAR AKTİVİTELER
DOSYALAR Dosya Yükle
Turuncu arka plan.png 1.9 MB - 13 Eylül, 16:14
Mor arka 2 MB - 13
ZAMAN TAKİBİ Başlat Ekle
1 sa 20 dk
1 sa 10 dk
2 s
ALT GÖREVLER Ekle
18:00
Belen And Partners
↳ Yazılımını kullanırken ihtiyaç duyduğumuz senaryolar hakkında sorularımız.
Müşteri Takibi +3
Bekliyor 75%

9:41
Mayıs

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

18:00
Belen And Partners
↳ Yazılımını kullanırken ihtiyaç duyduğumuz senaryolar hakkında sorularımız.
Fuat AYDIN
Müşteri Takibi +3
Aktif
18:00
Belen And Partners
↳ Yazılımını kullanırken ihtiyaç duyduğumuz senaryolar hakkında sorularımız.
Takvim

Figure 2: Screenshots of notifications, task details and calendar screens

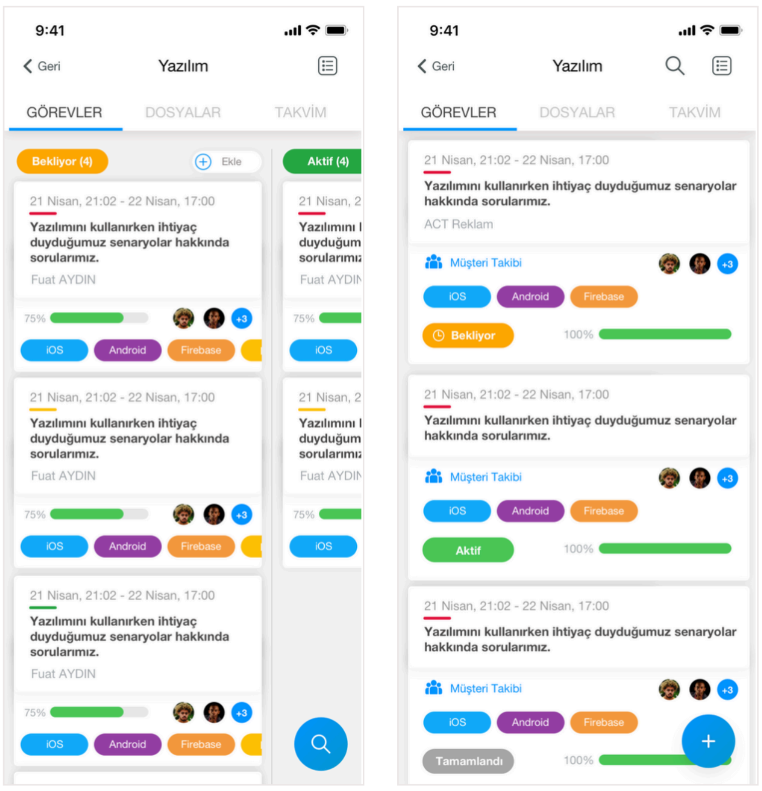
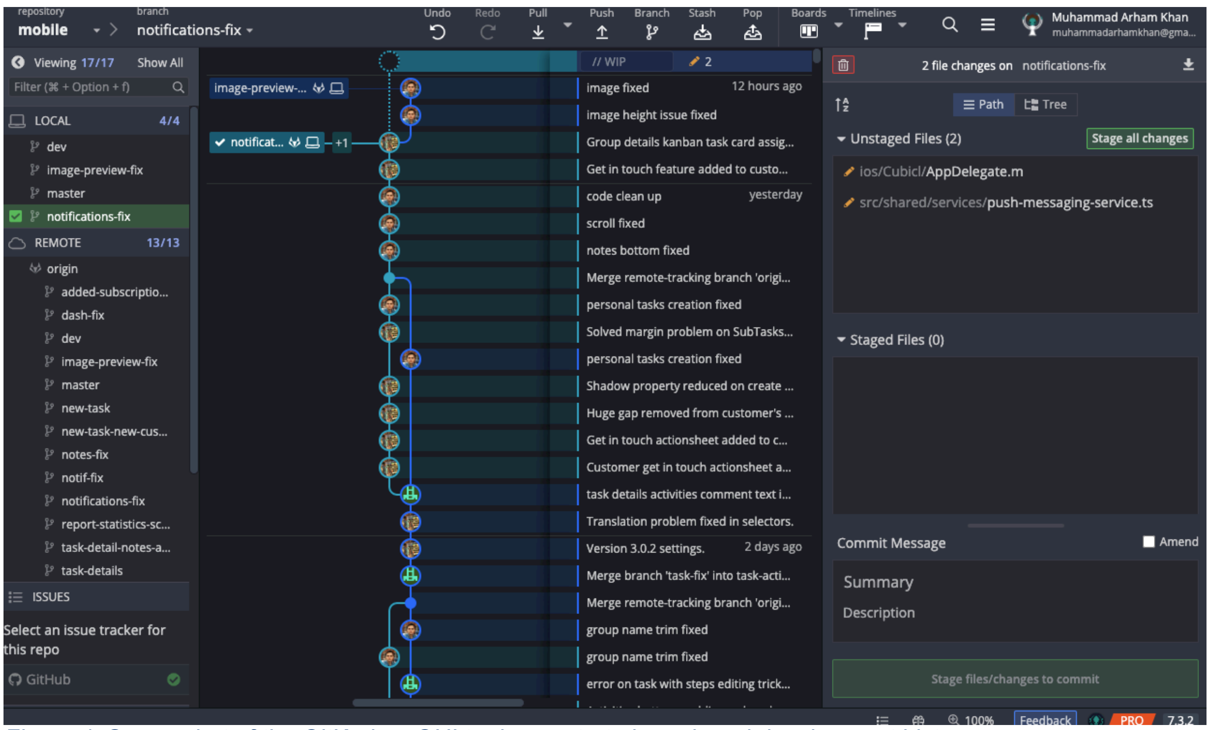


Figure 3: Screenshots of the group details screen in Kanban and list views



LOCAL: dev, image-preview-fix, master, notifications-fix

REMOTE: origin — added-subscriptio..., dash-fix, dev, image-preview-fix, master, new-task, new-task-new-cus..., notes-fix, notif-fix, notifications-fix, report-statistics-sc..., task-detail-notes-a..., task-details

Select an issue tracker for this repo

```
// WIP                                                   2
image fixed                                              12 hours ago
image height issue fixed
Group details kanban task card assig...
Get in touch feature added to custo...
code clean up                                            yesterday
scroll fixed
notes bottom fixed
Merge remote-tracking branch 'origi...
personal tasks creation fixed
Solved margin problem on SubTasks...
personal tasks creation fixed
Shadow property reduced on create ...
Huge gap removed from customer's ...
Get in touch actionsheet added to c...
Customer get in touch actionsheet a...
task details activities comment text i...
Translation problem fixed in selectors.
Version 3.0.2 settings.                                  2 days ago
Merge branch 'task-fix' into task-acti...
Merge remote-tracking branch 'origi...
group name trim fixed
group name trim fixed
error on task with steps editing trick...
```

2 file changes on notifications-fix

Unstaged Files (2)
- ios/Cubicl/AppDelegate.m
- src/shared/services/push-messaging-service.ts

Staged Files (0)

Commit Message

Figure 4: Screenshot of the GitKraken GUI to demonstrate branch and development history

Self-Checklist for Your Report

Please check the items here before submitting your report. This signed checklist should be the final page of your report.

- ✓ Did you provide detailed information about the work you did?
- ✓ Is supervisor information included?
- ✓ Did you use the Report Template to prepare your report, so that it has a cover page, the 7 major sections and 12 subsections specified in the Table of Contents, and uses the required section names?
- ✓ Did you follow the style guidelines?
- ✓ Does you report look professionally written?
- ✓ Does your report include all necessary References, and proper citations to them in the body?
- ✓ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Name: Muhammad Arham Khan