CmpE 451 Fall 2018

Group 10

First Customer Milestone Report

01.11.2018

Table Of Contents

1. EXECUTIVE SUMMARY

The purpose of Freelance Platform is providing an environment for freelancers and clients to collaborate in a single platform. In this way clients can post their tasks or projects regarding different subjects and deal with a professional after examining the results of bidding. Moreover, they can talk about further details about the project and handle payment issues via freelance platform in a secure way.

1.1 Summary of Project Status

Until now we mainly focused on our client's requests for first demo presentation. All our teams are in the same place in terms of functionality implementation. In backend, frontend and android a user can register, login, logout, edit his/her profile and post projects by declaring deadline, budget, project category and related tags. In addition to these, users can search projects by using tags, categories and project titles. Our backend and frontend are deployed on AWS. Our website can be accessed via http://35.178.179.18:8080.

1.2 Changes Planned

Currently, user can act as a client only. We are planning to add the freelancer role for the user. Also, corresponding changes will be done for the backend part. To illustrate, we expect a user to be able to select her role using a toggle in the main menu for android app and a toggle in the upper right section of the screen for the web app. After the selection made the user will be asked to complete the missing profile fields for that role if there are any. From that point the system will react to the user according to the selected role. With this feature the main structure of the project will be complete.

Upon the main structure some fundamental features will be added. Namely search filtering, bidding, rating, annotation, payment, recommendation engine features will be added. These critical features all have their places in the project plan ordered according to their urgence. Search is already a feature we have but we will make the user able to filter results by searching in categories/tags/etc. Bidding is the main concept of connecting freelancer and the project so this feature will have more thoughts on it. Rating will be

double-sided so that while freelancers being rated by their works clients will be rated for their employer skills. Annotations are planned only for images for now but if we decide on they will be useful in other fields like text we will be adding more onto this feature in the future. Payment will be done by a in-house currency for now. Recommendation engine will be a vital feature of the system when users that are searching through countless projects and trying to choose between many skilled freelancers are concerned.

2. DELIVERABLES

Name of deliverable	Status
Reviewed Requirements	Delivered
Project Plan	Delivered

Name of deliverable	Backend Status	Frontend Status	Android Status
Sign Up	Delivered	Delivered	Delivered
Login	Delivered	Delivered	Delivered
Logout	Delivered	Delivered	Delivered
Create Profile	Delivered	Delivered	Delivered
Post Project	Delivered	Delivered	Delivered
Search Project	Delivered	Delivered	Delivered

3. EVALUATION OF DELIVERABLES

All deliverables above mentioned are implemented and shown in the demo session. Our client gave us a feedback about fields of projects. According to their feedback, in future we will ask client users in our system to specify location of the project explicitly.

4. CODING WORK

Group Member	Work Done
Barış Can Esmer	Code review
Ahmet Faruk Çelimli	Backend user model, serializer Backend register api Backend login api Backend profile model, serializer Backend profile api Backend project model, serializer Backend project api Backend tag model, serializer Backend tag api Backend category model, serializer Backend category api Backend authentication and user permissions
Baran Kılıç	Frontend Sign up and Login Pages Frontend Homepage Frontend Profile and Profile Edit Pages Frontend Project View and Project Creation Pages Frontend Project Search Page
Canberk Yıldırım	Android Search Page Tab Layout
Kaan Özgen	Android Login Page Android Dashboard Android Auto Login Android Project Create Android Edit Profile Android Project Dialog Auto Suggest for tags in android
Elifnaz Utkan	Android Account Create Page
Yaşar Alim Türkmen	-
Muhammed Emin Vergili	Dockerization of the backend Dockerization of the frontend Docker-compose of the project Getting server up and making server settings Deployment of the project

5. REQUIREMENTS

Glossary

User	A person who opens and uses the application or website. Can be registered or unregistered.
Registered User	A registered user is a user of a website or mobile program who has previously signed up.
Guest User	A person who visits the application or website without signing in.
Freelancer	A user who takes projects in order to make money.
Client	A user who upload projects to system in order to be done by freelancers.
Bid	Offering a particular amount of money for a project.
Rating	An action which is done by clients and freelancers in order to evaluate both sides of the project.
Comment	Comprehensive evaluation that is done by clients and freelancers in order to give an idea to other users in the system about the client and freelancer.
Profile Page	A page that contains various information about a freelancer.
Tag	User-defined string which can be used to identify project or search for semantically related projects
Contract	A final agreement between the client and the freelancer after the client chooses a bid for the project. It demonstrates that the freelancer is still interested in the project.
Default Profile Segments	Profile parts a freelancer can change. These parts are name, image and introduction letter.

5.1. Functional Requirement

5.1.1 User Requirements

5.1.1.1 Login/Sign up Requirements

5.1.1.1.1 Users shall be able to sign up with their email, name and password.

5.1.1.1.2 Registered users shall be able to login with their email and password.

5.1.1.2 Freelancer Requirements

5.1.1.2.1 Freelancers shall be able to edit their profile page.

5.1.1.2.1.1 Freelancers shall be able to add and remove default profile segments.

5.1.1.2.1.2 Freelancers shall be able to choose the interest areas and tags they want to be notified of.

5.1.1.2.2 Freelancers shall be able to add a project to their favorites.

5.1.1.2.3 Freelancers shall be able to rate a project that they have completed.

5.1.1.2.4 Freelancers shall be able to pin and remove previous contracts to their account.

5.1.1.2.5 Freelancers shall be able to filter search results down to their certain categories, tags, and budget range.

5.1.1.2.6 Freelancers shall be able to bid for a project.

5.1.1.2.7 Freelancers shall be able to inform clients about project completion by using messaging service.

5.1.1.2.8 Freelancers shall be able to upload their works to system.

5.1.1.3 Client Requirements

5.1.1.3.1 Clients shall be able to create project postings.

5.1.1.3.1.1 Clients shall be able to set a deadline for their project postings.

5.1.1.3.1.2 Clients shall be able to add tags to their project postings.

5.1.1.3.1.2.1 Clients shall be able to choose tags from a searchable list of preexisting tags and create new tags(not used before) for their project postings.

5.1.1.3.1.3 Clients shall be able to choose a budget range for their project postings.

5.1.1.3.1.4 Clients shall be able to add project details to their postings using a rich text editor.

5.1.1.3.2 Clients shall be able to propose a project to a certain freelancer.

5.1.1.3.3 Clients shall be able to decide on any bid placed on their project.

5.1.1.3.4 Clients shall be able to delete a project.

5.1.1.3.5 Clients shall be able to contract any freelancer who placed a bid on the project.

5.1.1.3.6 Clients shall be able to cancel a contract if freelancer misses the deadline.

5.1.1.4 Feedback Requirements

5.1.1.4.1 Users shall be able to comment and post feedback on the project.

5.1.1.4.1.1 Clients shall be able to post feedback for working process, final product and also a text input for detailed elaboration.

5.1.1.4.1.2 Freelancers shall be able to post feedback for payment, communication, project specifications/description.

5.1.1.4.2 Users shall be able to rate and comment about users they worked with.

5.1.1.5 User Profile Requirements

5.1.1.5.1 Users shall be able to view public profiles of other freelancers and clients.

5.1.1.5.2 Users shall be able to pin feedbacks to top of their profile.

5.1.1.5.3 Users shall be able to enable notifications through e-mail to get notified of messages and actions related to them.

5.1.1.5.4 Users shall be provided a way of account recovery option in case they lose access to their account.

5.1.1.6 Annotation Requirements

5.1.1.6.1 Users shall be able to annotate the text in the project description.

5.1.1.6.2 Users shall be able to annotate the images in the project description.

5.1.1.6.3 Owner user of an annotation shall be able to edit or delete his/her annotation.

5.1.2 System Requirements

5.1.2.1 Bidding Requirements

5.1.2.1.1 Given bids to a project with their amounts and given dates shall not be visible to any user except the client.

5.1.2.1.2 The system shall notify the users upon a change in the post details.

5.1.2.1.3 In case of a change to post details, the post shall be suspended temporarily and be put in moderation queue to confirm the change.

5.1.2.2 Recommendation Engine Requirements

5.1.2.2.1 Freelancers shall be given a recommendation of project postings that are related to their previous jobs, fields, interests and tags they prefer.

5.1.2.2.2 Clients shall be suggested of top and rising freelancers that may be interested in their posting.

5.1.2.2.3 During project posting creation, a list of successfully completed projects of similar topics shall be presented to the user to help them better specify the scope of the project.

5.1.2.3 Payment Requirements

5.1.2.3.1 The advance payment that the client promises shall be withdrawn from their account and be held until both parties agree that the conditions to said payment is met.

5.1.2.3.2 Client shall not be able to contract with a freelancer if he/she doesn't have enough cash in his/her account.

5.1.2.4 Search Requirements

5.1.2.4.1 The system shall support both basic text and semantic search using the keywords user has typed in.

5.1.2.4.2 The system shall provide filtering mechanism by considering the title and description of the contents.

5.1.2.5 User Creation Requirements

5.1.2.5.1 The system shall suggest user to create an account only when they want to commit to creating a project posting.

5.2. Nonfunctional Requirements

5.2.1. Security Requirements

5.2.1.1 The passwords stored in the database of the system shall be encrypted.

5.2.2. Reliability Requirements

5.2.2.1 The service shall be implemented with scalability in mind. It shall be able to gracefully handle bursts of user activity using horizontal scaling.

5.2.3. Performance Requirements

5.2.3.1 The service shall respond to any operation within two seconds.

5.2.4. Usability Requirements

5.2.4.1 All string templates used throughout the service shall be translatable.

5.2.5. Accessibility Requirements

5.2.5.1 The service shall comply with Web Content Accessibility Guidelines proposed by W3C.

5.2.6. Annotation Requirements

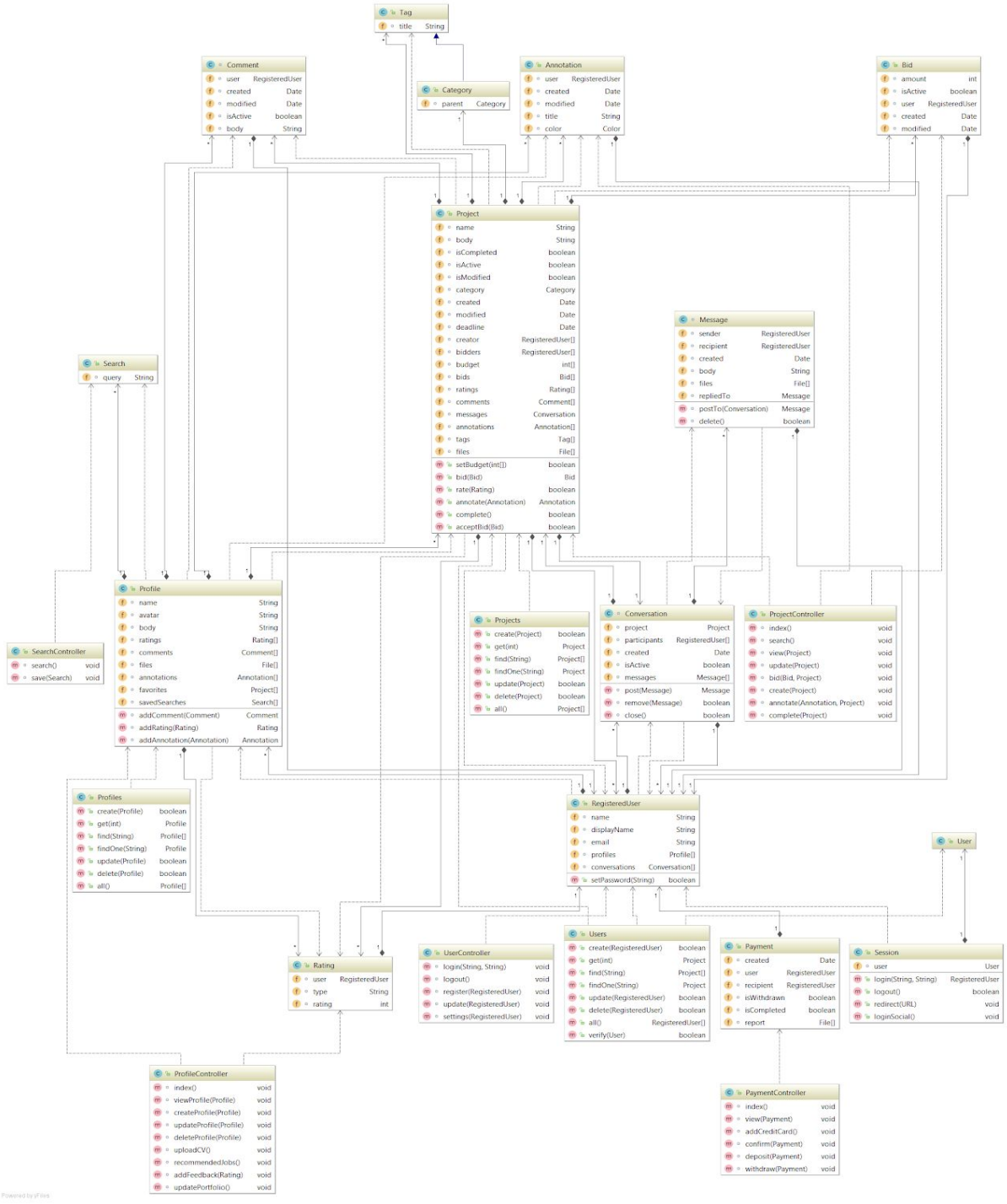
5.2.6.1 The system shall comply with W3C Open Annotation Standards.

5.2.7. Availability Requirements

5.2.7.1 The Android application shall support Android 5.0 or upper versions (70% of android devices can be included).

5.2.7.2 The system shall have a web application which supports the latest version of web browsers.

6. Design


Tag
title String
Comment
user RegisteredUser
created Date
modified Date
isActive boolean
body String
Category
parent Category
Annotation
user RegisteredUser
created Date
modified Date
title String
color Color
Bid
amount int
isActive boolean
user RegisteredUser
created Date
modified Date
Project
name String
body String
isCompleted boolean
isActive boolean
isModified boolean
category Category
created Date
modified Date
deadline Date
creator RegisteredUser[]
bidders RegisteredUser[]
budget int[]
bids Bid[]
ratings Rating[]
comments Comment[]
messages Conversation
annotations Annotation[]
tags Tag[]
files File[]
setBudget(int[]) boolean
bid(Bid) Bid
rate(Rating) boolean
annotate(Annotation) Annotation
complete() boolean
acceptBid(Bid) boolean
Message
sender RegisteredUser
recipient RegisteredUser
created Date
body String
files File[]
repliedTo Message
postTo(Conversation) Message
delete() boolean
Search
query String
Profile
name String
avatar String
body String
ratings Rating[]
comments Comment[]
files File[]
annotations Annotation[]
favorites Project[]
savedSearches Search[]
addComment(Comment) Comment
addRating(Rating) Rating
addAnnotation(Annotation) Annotation
SearchController
search() void
save(Search) void
Projects
create(Project) boolean
get(int) Project
find(String) Project[]
findOne(String) Project
update(Project) boolean
delete(Project) boolean
all() Project[]
Conversation
project Project
participants RegisteredUser[]
created Date
isActive boolean
messages Message[]
post(Message) Message
remove(Message) boolean
close() boolean
ProjectController
index() void
search() void
view(Project) void
update(Project) void
bid(Bid, Project) void
create(Project) void
annotate(Annotation, Project) void
complete(Project) void
Profiles
create(Profile) boolean
get(int) Profile
find(String) Profile[]
findOne(String) Profile
update(Profile) boolean
delete(Profile) boolean
all() Profile[]
RegisteredUser
name String
displayName String
email String
profiles Profile[]
conversations Conversation[]
setPassword(String) boolean
User
Rating
user RegisteredUser
type String
rating int
UserController
login(String, String) void
logout() void
register(RegisteredUser) void
update(RegisteredUser) void
settings(RegisteredUser) void
Users
create(RegisteredUser) boolean
get(int) Project
find(String) Project[]
findOne(String) Project
update(RegisteredUser) boolean
delete(RegisteredUser) boolean
all() RegisteredUser[]
verify(User) boolean
Payment
created Date
user RegisteredUser
recipient RegisteredUser
isWithdrawn boolean
isCompleted boolean
report File[]
Session
user User
login(String, String) RegisteredUser
logout() boolean
redirect(URL) void
loginSocial() void
ProfileController
index() void
viewProfile(Profile) void
createProfile(Profile) void
updateProfile(Profile) void
deleteProfile(Profile) void
uploadCV() void
recommendedJobs() void
addFeedback(Rating) void
updatePortfolio() void
PaymentController
index() void
view(Payment) void
addCreditCard() void
confirm(Payment) void
deposit(Payment) void
withdraw(Payment) void

7. Project Plan

Backend	**2018-10-01**	**2018-12-28**
API Blueprint	2018-10-01	2018-10-08
Tech Decisions(Framework, IDE, Database)	2018-10-01	2018-10-08
Runnable Hello World	2018-10-01	2018-10-08
Structuring Project	2018-10-09	2018-10-16
DB primitive tables	2018-10-09	2018-10-16
Mock Data	2018-10-09	2018-10-16
Deployment Process	2018-10-17	2018-10-24
Environment/Branches	2018-10-17	2018-10-24
Test Implementation-1	2018-10-25	2018-11-01
Basic API services	2018-11-02	2018-11-08
Milestone 1	2018-11-09	2018-11-09
Search Filtering	2018-11-10	2018-11-16
Recommendation Engine	2018-11-17	2018-11-23
Payment Integration	2018-11-24	2018-11-30
Project Review	2018-12-01	2018-12-06
Milestone 2	2018-12-07	2018-12-07
Rating System	2018-12-08	2018-12-14
Annotation System	2018-12-15	2018-12-21
Notification Services	2018-12-22	2018-12-28
Frontend	**2018-10-01**	**2018-12-28**
API Blueprint review	2018-10-01	2018-10-08
Tech Decisions	2018-10-01	2018-10-08
Home Page Hello World	2018-10-01	2018-10-08
Login and Register Pages	2018-10-09	2018-10-16
Profile Creation	2018-10-09	2018-10-16
User/Client	2018-10-17	2018-10-24
Profile Pages	2018-10-17	2018-10-24
Project Creation	2018-10-25	2018-11-01
Bidding Page	2018-10-25	2018-11-01
Search Page	2018-11-02	2018-11-08
Dashboard	2018-11-02	2018-11-08
Milestone 1	2018-11-09	2018-11-09
Search Filtering	2018-11-10	2018-11-16

Recommendation Engine	2018-11-17	2018-11-23
Payment Integration	2018-11-24	2018-11-30
Project Review	2018-12-01	2018-12-06
Milestone 2	2018-12-07	2018-12-07
Rating System	2018-12-08	2018-12-14
Annotation System	2018-12-15	2018-12-21
Notification Services	2018-12-22	2018-12-28
Android	**2018-10-01**	**2018-12-28**
API Blueprint review	2018-10-01	2018-10-08
Tech Decisions	2018-10-01	2018-10-08
Home Page Hello World	2018-10-01	2018-10-08
Login and Register Pages	2018-10-09	2018-10-16
Profile Creation	2018-10-09	2018-10-16
User/Client	2018-10-17	2018-10-24
Profile Pages	2018-10-17	2018-10-24
Project Creation	2018-10-25	2018-11-01
Bidding Page	2018-10-25	2018-11-01
Search Page	2018-11-02	2018-11-08
Dashboard	2018-11-02	2018-11-08
Milestone 1	2018-11-09	2018-11-09
Search Filtering	2018-11-10	2018-11-16
Recommendation Engine	2018-11-17	2018-11-23
Payment Integration	2018-11-24	2018-11-30
Project Review	2018-12-01	2018-12-06
Milestone 2	2018-12-07	2018-12-07
Rating System	2018-12-08	2018-12-14
Annotation System	2018-12-15	2018-12-21
Notification Services	2018-12-22	2018-12-28

8. Code

For each feature, we create a separate branch and follow the branch naming: <platform>-<feature-name> where platform is backend, frontend or android. Also, if we fix a bug, we append “-bugfix” suffix to our branch name. Some examples from our repository are

“frontend-project-creation”, “backend-project-category-bugfix”, “backend-project-api”. Instead of merging branches directly, we use pull requests. It gives us an opportunity to review our code and document the changes in the code. In addition, we can add labels to our pull requests and this allows us to categorize the pull requests easily. Pull requests are a communication channel and informs others about our work and progress.

9. Evaluation of Tools and Managing the Project

9.1 teamgantt

We use teamgantt in order to prepare our project plan. We had used ProjectLibre for this task last semester and we can easily say that teamgantt is a better solution. It is easier to use and produce a good looking plan.

9.2 Slack

Slack is good for communication. It has mobile apps, website clients and desktops clients so we can use it on every platform. If we need to discuss a separate topic, we can create a new channel and add the appropriate people to this channel. This enables us to separate a discussion from another. In addition, Slack supports apps. We can integrate Github to Slack.

9.3 Trello

Trello helps us to organize our tasks. Thanks to Trello every member of subteams(backend, frontend, android) can see which are the tasks should be implemented and which of them are currently is on progress.

9.4 Git

Not need to explain Git. It is excellent. It makes version control easy and enables us to work collaboratively. We can create branches, work on different features in parallel and merge them to our master branch.

9.5 Github

Github extends Git. In Github, we can have issues. Issues are useful to track bugs but we also used it for task management. We can filter issues and add tags to them. This makes easy to find an issue when we need it. We can comment on issues. This improves communication in the team. We can create wiki using markdown. With wiki, we can document design, plans and meetings of our project. In addition, we can make pull request and assign reviewers to this pull request. This reviewing mechanism improves the quality of the resulting code since people cannot see their own mistakes easily but see others mistakes easily.

9.6 PyCharm

PyCharm is an IDE that we used in order to implement our backend. It provides smart code completion, on-the-fly error highlighting. It checks the code according to PEP8 style guide. Therefore, all team members follow the same standard.

9.7 Django

We use django framework to implement our backend. It provides many built-in functions which helps to implement faster. It creates migration files and by using them we can create database tables by just writing a command. These migration files also enable developers to switch between different databases easily.

9.8 Django Rest Framework

We use django rest framework to implement our APIs. It provides a nice interface to examine inputs and outputs of each API. In this way, all members of our team can test APIs without using a third party application like Postman. Rest framework provides built-in authentication and token functions. The framework also has built-in search modules which helps us a lot to implement our search APIs.

9.9 Vue.js

We chose Vue.js framework to implement our frontend. It is the third popular frontend framework (after React and Angular). It is simple and has a gentle learning curve. It uses HTML, CSS and JS. You do not need to learn Typescript (like in Angular). Vue.js is lightweight. It is only 20KB when minimized and gzipped. Vue.js is also fast like its rivals because it utilizes a virtual DOM. It has a template system similar to Angular. It is easy to read and write the template code.

9.10 Bootstrap

We chose Bootstrap framework for UI. It is the most popular UI framework. It is easy to learn and has an extensive documentation and examples. It has a grid system that makes building responsive web pages easier. The main drawback of Bootstrap is that every website that uses it looks alike. Also, the default color palette of Bootstrap has low color contrast. (a bad trend in today's web design) This decreases readability and accessibility.
Instead of directly using Bootstrap, we used BootstrapVue framework since it integrates better with Vue and implements the Javascript parts of the Bootstrap in Vue (otherwise Vue could not detect the changes in Bootstrap elements since Vue uses a virtual DOM).

9.11 Android Studio

Android studio is chosen for android side of our project since it provides default emulator and a lot of integration tools. Apart from that, using this IDE helps us develop android app fast and see where we get errors or warnings, it provides code completion, on-the-fly error highlighting as well.

9.12 Retrofit API

Using Retrofit API for android project was convenient because our backend team is using Django.And Retrofit calls are very similar to it and easy to implement.