Summer Practice Report

Middle East Technical University Computer Engineering Department

CENG300

Name: Göktuğ Ekinci
Student ID: 2380343
Year: 2
Start Date: 4^{th} of August, 2020
End Date: 14^{th} of September, 2020

Supervised By
Assoc. Prof. İsmail Sengör Altıngövde

Contents

1 Introduction

1.1 General

I have done my summer internship regarding the course of CENG300 in JotForm and have performed my internship remotely. JotForm is a form builder company which has both website, which is www.jotform.com, and mobile application. There were a lot of units that accepts interns in the company such as user research, data science and web design. I applied for the data science unit. It was a 5 week program, and my studies were supervised full time data scientist. Company's communication were run on Discord channel. There were plenty of channel that represents team/units and another channel for interns and teams of interns. (Data science intern, user experience channel..) I were expected to start working and join the discord channel's voice channel by 8-10 AM and finish 5-7 PM. (9 work hours expected) Also there were special days in weeks, which are Poker Day, Happy Hour, and Demodays. At poker days, we were playing poker for fun online with the people. At Happy Hour, we were chatting with people in divided groups and playing different kind of games. We were using Zoom during the Happy Hour. I am going to talk about the DemoDay in detail in the further pages of my report.

1.2 Schedule of the Summer Internship Program

Internship was divided into 5 weeks:

1.2.1 First Week

This week was a warm-up week. All interns of all units were given simple form building tasks.

1.2.2 Second Week

From the second week. Interns were appointed a mentor, and choose their projects from the project pool.

1.2.3 Third&Fourth Week

These weeks interns worked on their projects with the help of their mentors.

1.2.4 Fifth Week

This week interns are expected to finish and polish their work, and the last day of the last week. There were Demoday's at the end of every week. At Demoday, teams are presenting what they've done last week, and at the end of the demos, interns that have completed their internships present their work on their projects. End of the fifth week I was expected to perform a presentation.

1.3 Mentor

My works are mentored and supervised by Soner Durmaz, Data Sciencist at JotForm and also graduate from our department.

2 Organization

2.1 General Information

JotForm is online form building organization which located in San Francisco. JotForm's product makes forms with a simplified creation instrument and an alternative to encode client information. Starting at 2015, JotForm had 2 million enlisted clients and 7,000 form templates. The forms can be incorporated on outer locales including MailChimp, PayPal, Stripe, and Salesforce. JotForm was established by Aytekin Tank at 2006.

2.2 Name&Website

JotForm
www.jotform.com

2.3 Location

San Francisco, California, U.S.

2.4 Structure

Company is divided into teams that work separate and different from each other. Each team has distinct job to do, they choose their own name such as Avengers Team, Patlıcan Team. Teams work in different areas such as

2.4.1 Data Science

This team's job is to manipulate and store the data in vary structures. Get the desired statistics directly, and also use AI&ML to build tools for form building.

2.4.2 User Experience

This team is responsible from reaching users of JotForm, receiving feedback and learning what people like or what to bring to JotForm.

2.4.3 Web Design

Web Design, whose job is to implement built structures or algorithms to the website. This team is also responsible from the user interface of the website.

2.4.4 Maintenance

This team is responsible for the general maintenance and server status of the site. Solving bugs is also this team's job.

3 Internship

In this section I divided my weeks in the internship periods into subsections. I have worked on three projects in these five weeks. I started working on projects after the first week.

3.1 First Week

3.1.1 Introduction of the Week

This week I was given five tasks to complete during the all first week. Tasks were about different kinds of form building. The aim is to warm-up, learn the form builder's tools, and how they work. I completed all the tasks of all week in one day. I asked for extra tasks and finished them this week too.

3.1.2 Tasks

Rules: - Please do not use any form templates; we expect you to create forms from scratch.
- The language of the forms should be English.

3.1.2.1 TASK 1.1 Create the form described below
A photography request form that asks for the necessary information for the photoshoot process. The final price of photography should change according to the customer's requirements. When the order is complete, the customer will receive an email that includes a well-designed PDF document of photography information.

3.1.2.2 TASK 1.2 Create the form described below
A university uses forms to manage course registration processes. When a student selects his/her department and courses, a mail containing related information is being sent to the corresponding department. For Example:
The student is studying in the Computer Engineering department. He is taking C++, Java, software management courses from the Computer Engineering department. After the registration process is done, a mail is sent to computerengineering@university.com. Also, the student receives an email about the courses he registered. The same form is also used by the sociology department when a student from the sociology department register for a new course (For Example, Modern Communication Systems, Criminology, Urban Poverty) through the form. An email will be sent to sociology@university.com and also to the student. Please create a form that applies to this case.

3.1.2.3 TASK 1.3 Create the form described below
Create a form that includes two different payment integrations on it.

3.1.2.4 TASK 1.4 Create the form described below
Create a quiz form and show different thank you pages based on the user's score.

3.1.2.5 TASK 1.5 Create the form described below
A Job application form with a custom autoresponder.

3.1.2.6 TASK 1.6 Create the form described below
Create a form displayed in 3 languages according to selected languages, show thank-you pages, and receive the notification email according to choose languages.

3.1.2.7 TASK 2.1 & TASK2.2 Create two forms using at least one integration in each. You are expected to be very creative at this point. We are aware it is straightforward to create any form that has an integration. However, this task aims to make you think of a creative user scenario that can be accomplished with JotForm&its integrations. For Example, this one is a good scenario: Using Slack integration, when a feedback form is filled, the support channel on slack can receive a message with the content of the feedback. Additionally, the correct person can be tagged automatically so that she/he will be notified when form filler selects his/her department while filling the form. You are expected to implement such forms.

3.1.2.8 TASK 3 Create a form that uses IFTTT / ZAPIER or webhook integration. Again you are expected to be creative while creating a solution in this form.

3.1.2.9 TASK 4.1 & TASK 4.2 Create two forms that use different widgets. You can check available widgets here.

3.1.2.10 TASK 5 Pick one of the options below:
- Redesign any of the two form templates you pick
- Use Google to find what kind of forms are needed by Education professionals. Find 10 PDF or hard copy form links (scanned images of paper forms) such that if we create them and post them in the form templates gallery of JotForm, we can attract education professionals searching for these forms.
- Consider you are expected to prepare web pages for every 81 cities in Turkey. Prepare a contact form for each of them.

All forms are expected to have the following fields:
Label: Shows Name of the city

Name:
Surname:
Email Address:
Phone Number:
Note:

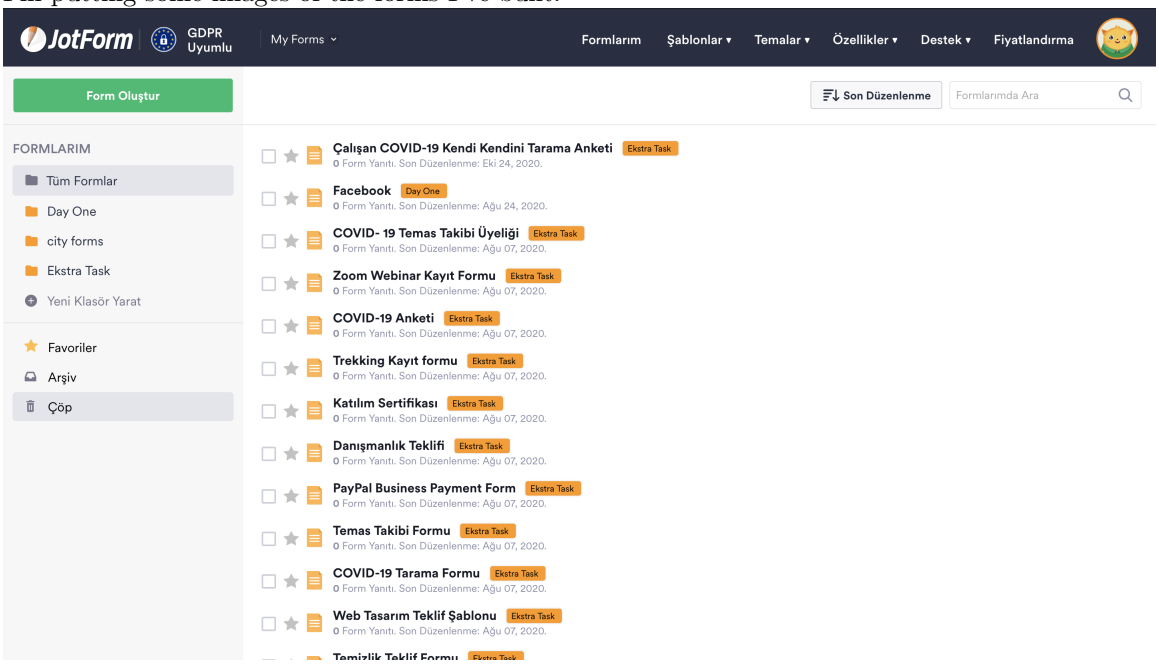
3.1.3 Work

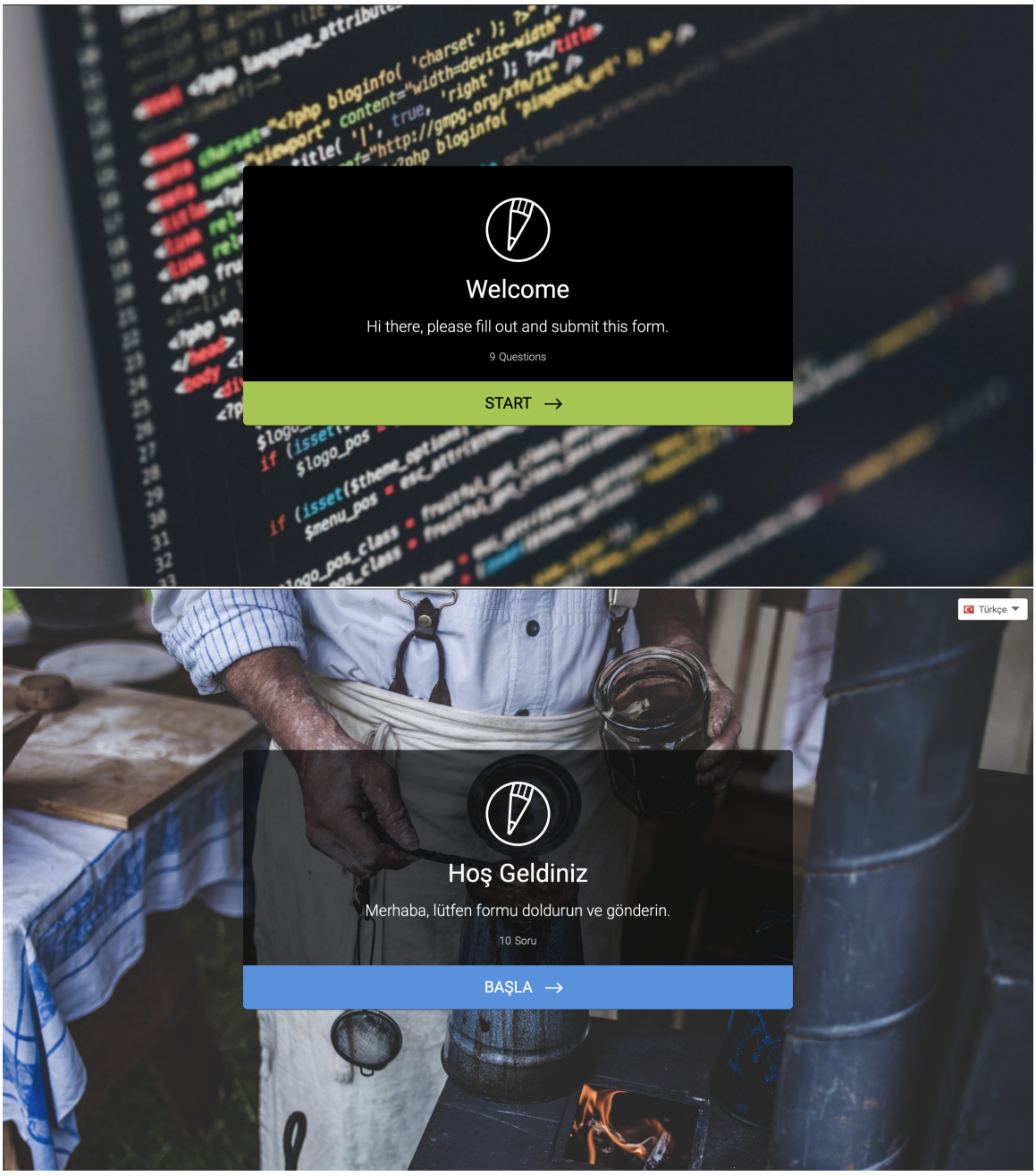
After I finished all the tasks and asked for extra tasks using JotForm's builder, my temporary mentor told me to complete the other options of the fifth task and make the Turkish translation of the 25 forms.

3.1.3.1 JotForm API One of the other options (Building forms for all the cities in Turkey) of the fifth task required too much effort for brute force. So, I decided to use the API. A Python library of JotForm's API, a library for Python 2, and not updated, probably not used long. When I tried to use the API, which was a default usage, I did not request a special task from API; it gave me an error. I started looking for what might have happened and realized that a library called urllib2 was getting no support from the developers when urllib2 had a chance to give errors when sending a request to a site is all about the library. Developers of the library are not solving this error due to security problems. To sum up, the library was not working in python2. I offered the temporary mentor to update the library; he said they'll let me know when there is an initiation. So I decided to use API's website and created 81 forms, and completed the task. API's website: *api.jotform.com/docs*

3.1.4 Results

I'm putting some images of the forms I've built.





3.2 Second Week

3.2.1 Introduction of the Week

This week I was assigned a mentor, Soner Durmaz, and I am asked to choose a project from the project pool of JotForm internship projects.

3.2.2 Available Projects

The project list was like this:

Data Science Intern Projects

1- Creating dedicated form submission reporting for specific form fields where our reporting tool is lacking. New chart views for each of them with regards to data type each of them collects.

2- Detect most used widgets and determine specific chart views for each of them on submission reporting.

3- Analyze all JotForm dashboards and create new visualization for all metrics to make them more readable.

4- Form categorization with machine learning algorithms

5- User industry Categorization with machine learning algorithms

6- Detecting most landed answers, blogs, and help pages and analyzed their content. Determine their common point as keyword groups and see what gets the attention of organic visits.

7- Detect most landed Form Template pages and analyze their content. Determine their common point as keyword groups and see what gets the attention of organic visits.

8- JotForm competitor research: Gather all competitor landings and analyze their content. Detect their main focus and report where JotForm is lacking.

9- Onboarding research: Compile all JotForm features and an average percentage of their usage in specific periods. Try finding a correlation if using features has anything to do with subscribing to JotForm.

10- Advertising comparison to competitors: Compile all search terms which create traffic on JotForm and our competitors. Analyze where JotForm can improve.
11- Analyze the JotForm support forum and all its content. Determine the main issues with JotForm and where all user requests are focused on.

12- Create a machine learning algorithm that works on support forums and suggests related forum threads upon submitting a new question. Users should be able to navigate through the associated parts of the forum before they receive an actual response to their problems.

13- Analyze all external posts of JotForm on different sources like Medium, Entrepreneur, etc. Determine their success, standard points, what they have provided us so far, and their contribution to our branding with our spending is taken into consideration.

14- Create an email user list compiler, which should detect specific user segments and end up with user list csv. Get help from all the user request lists forwarded to the Data Team so far.

15- Detect must have questions on specific form types to notify users with field suggestions.

16- Building drag and drop reporting tools like Google Data Studio: Data can be gathered from CSV, Excel, or DB. Users should select which column to visualize, and the system should provide a meaningful chart type based on the selected column type. The user should also be able to drag and drop columns to draw charts. It should remove multiple series/lines, and it should automatically scale lines according to the data levels (hint: adding various axes).

17- Creating an insight tool: The system should predict an insight according to previous data and send warnings for negative or positive situations. For Example, if something's wrong with our signup numbers according to the understanding, the system should warn before it's too late. This project can use our daily

stats data or P30 data for prediction.

18- A language-translation tool: By using different neural network approaches, a language transformation tool which will work on English-German and English-Spanish translation.

19- Background Image Generation: A text-to-image generation using Generative Adversarial Networks(GAN) and generating a deep learning model that creates background images, given the input keywords as input.

20- JotForm trends: build a tool using the words on user forms. This should work like a trend analyzer and should be able to tell us the trending form keywords of a specific time frame compared to a past time frame. We should also be able to see how a particular keyword's usage changes over time. Use the form creation dates to link time with keywords.

21- Build a Recommendation Engine for JotForm widgets; if a user adds a specific device to a form, we can recommend new widgets based on the initial selection

22- Build a phishing detector: create a phishing detector that can detect spammers that use our forms to steal identity information or such. Make sure to function in several languages.

3.2.3 First Project Description

I chose project number 19, Background Image Generation, with GAN. I chose this project since I do not know anything about Generative Adversarial Network's and think I can learn during the internship. This project aims to create a background image for the forms using the textual data users entered.

3.2.4 Preliminary Work & Literature Research

I did not know whether anybody worked on this project or not. When I asked my mentor, he told me there is no study for this project before and ask me to start from scratch. This week, I did literature research about GAN and TensorFlow. Tensorflow was a bit necessary since learning a new library was tempting, and nearly every resource about GAN was using TensorFlow instead of PyTorch; I knew PyTorch but not TensorFlow. I watched TensorFlow tutorials and searched about Generative Adversarial Networks. If I give a brief intro to GAN;

3.2.4.1 Generative Adversarial Network(GAN) GAN is a network for image generation usage and uses two cooperating networks to generate these images. These two parts are Discriminator and Generator.

3.2.4.2 Discriminator The discriminator is a neural network for image classification. It classifies the images, whether it is the real image or a fake image of the wanted object. It is trained by the wanted pictures of the wanted items, and it learns which is our object and which is not.

3.2.4.3 Generator The generator is the first stage of the GAN. The generator is a Neural Network for image generation. The results of the discriminator train it. If it creates a real wanted image, it is a plus for it, but if it does not, the discriminator punishes and does not let the image be an output.

3.3 Third Week

3.3.1 Dateset Inspection

The dataset consists of 30 000 lines of input. These lines contain two columns. While the first column is formed of textual input, the second column contains links of the background images.

3.3.1.1 Textual Data This textual data comes from the form question headers or any kind of description written by the builder of the specific form. So, that means data can contain countless words, expressions, or any kind of individual text. Also, almost all of the input text has an Html code at the end of the text. That brings us to very complicated inputs. Here is some example of inputs;

3.3.1.2 Images Images contain visual background for the specific form. These background images are selected by the users, which means photos can come from everywhere, or it can be photo contains anything. That was a problematic point because even image generation itself is a complicated and hard process. Completely random images make this task a lot way complicates the process. Here is some input images;

3.3.2 Method

One of the ideas that came to my mind was generating images with GAN and getting a text from that image; some tools can infer a word or text from an image and see if it fits the desired form.
Another one was categorizing the dataset into form types such as registration, application, and payment.

3.3.3 Technical Problems

3.3.3.1 Contradiction with GAN GAN was designed to produce an object or a being but not more, but I was expected to produce photos of more than one type of object in this project. For Example, if there is money in the text and it is a payment form, I was expected to produce an image contains money, but this topic of the photo part can change from form to form. That was a challenging problem, and it made me start thinking about the feasibility of this project with a GAN solution.

3.3.3.2 Dataset Problems There were lots of problems with the dataset. The first problem was that the dataset had repetitive lines, and the number of these repetitive lines was not underestimating. Nearly more than half of this dataset was containing the same lines again. The other problem with the dataset is that the textual part of the inputs was completely random. It can contain countless words or just one word; this was completely random.

The biggest problem with the dataset, which affects the feasibility of this problem much, is that textual data and visual data do not have to have a correlation. For Example, let's say a pizza Restaurant prepared a form that accepts the order. This Restaurant is not required to put a pizza or any kind of food photo for the background. They can put a flower photo for a relaxing background, and when it comes to training the neural network, it is nearly impossible for us to expect the neural network to be trained. Datasets itself does not hold a correlation; so, we can not expect a neural network to find a correlation. Here are some examples

of images which images are in the same clusters, Registration Forms.

Here is a woman staring at the camera. This background is from a fitness club's registration form. Let's look at another background image of another fitness club.

As can be seen, there is any correlation between the two images.

3.3.4 Report

I decided to prepare a report, I thought this was an appropriate and a more formal way to communicate. The aim was to present that the project has no feasibility with GAN. After reaching him with the report, the mentor agreed with me and said I could pass to another project.

3.4 Fourth Week

3.4.1 New Project

The second project that I contributed was number 21 from the project list, which is 'Widget Recommendation for Form Builder'. I chose this project since I had no experience with recommendation systems before to learn and felt the lack of widget recommendation in the first week when struggling with form building. This project aims to recommend widgets to a user who used one or more widgets in their form.

Widgets are used in the Form Builder to improve the form and add different capabilities to the form. One of the widgets is a calculation tool that can get the numeric answers of the questions and do the calculation with these answers, which the user can manage the process. Another widget is an E-Signature widget that provides e-signature support for the form. There are 250 to 300 widgets in JotForm.

3.4.2 Dataset Inspection

Dataset consists of 322 643 lines of input. These lines contain 1 column, and that column is formed of widget names separated by columns. Every line represents a form, and the widgets in the line the widget used by the user for that form. Dataset is relatively simple compared to the previous project. The number of widgets can vary from one to another. Here are some examples of data;

0- 'Birth Date Picker, Bootstrap Switch Field, Cities of Indonesia, Email Validator, Field Multiplier, Form Tabs, Global Countdown, Image Picker, Mobile Responsive, Phone Picker, Progress Bar, Show Map Location, Smooth Signature, Take Photo, Terms & Conditions, YouTube'

1- E-Signature, Take Photo

2- 'Auto Complete,Barcode Scanner,Bootstrap Switch Field,Checkbox in Dropdown,Code Editor,Day Countdown,Digital Clock,Disqus,Double the Donation,Draw on Image,E-mail Validator,Email Correctness Check,Facebook Comments,Fancy Radios,Fancy Timer,Get Form Page URL,Gift Registry,Global Countdown,Kinomap,Like and Dislike,Livestream,Mini Date Picker,Minimal Radio Buttons,Pastebin,Phone Picker,Polaris Radio
Buttons,Socialcam,Square Radio Buttons,Squire Editor,TextArea Autosize,Twitch,Unique ID,Voice Recorder,XVerify Email,Youku,YouTube'

3- 'E-Signature,Geo Stamp,GeoComplete,Geolocation,Get Visitor Location,GPS Location,Image Checkboxes,Kinomap,Location
Coordinates,Photozou,Take Photo,Unique ID'

4- Form Calculation, Iframe Embed, Show Map Location, Squire Editor'

322638- QR Code Reader, YouTube

322639- Birth Date Picker, Smooth Signature

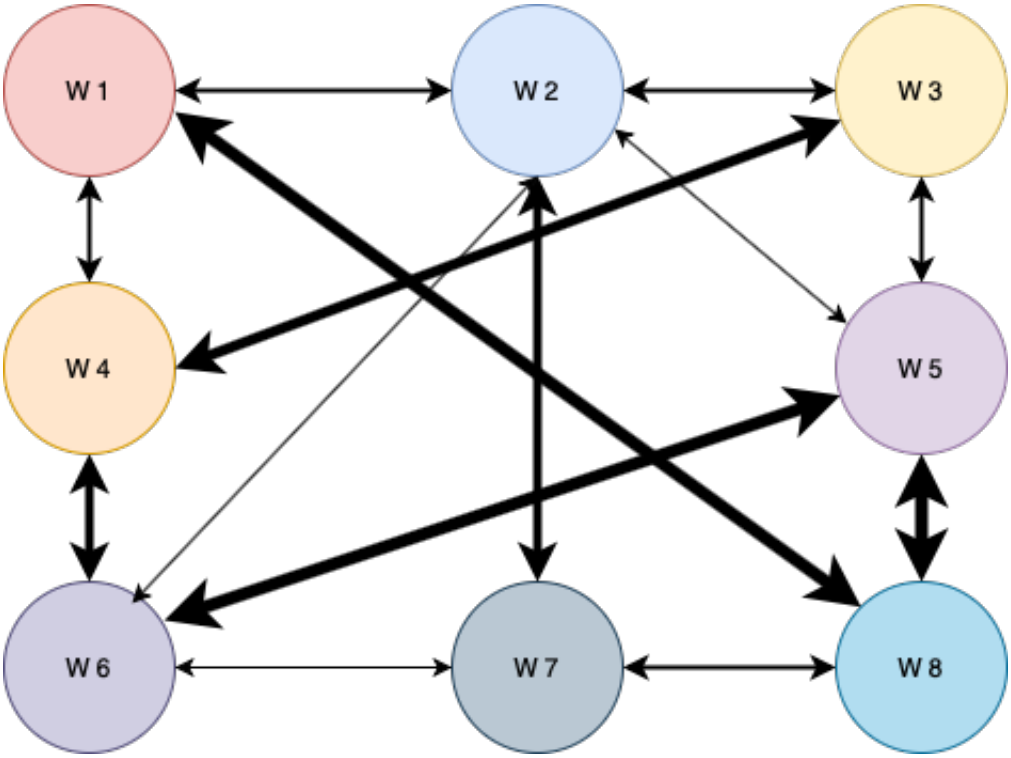
322640- Dates Difference,Dropdown with Paging,Dynamic ...

322641- E-Signature,Smooth Signature,Terms & Conditions

322642- Mobile Responsive,Terms & Conditions

3.4.3 Idea&Solution

3.4.3.1 Graph My approach to this problem was a graph-based solution. If two widgets are used in the same form, they are correlated and connected. I decided to add every individual widget to the graph as a node and put edges to two nodes when they are used in the same form. In this way, the aim was to build a large graph containing every widget as nodes and correlations between nodes as edges.



This diagram represents a graph where the circles represent widgets, and the connections represent edges. The thickness of the edge implies the strength of that edge; in other words, thickness represents how much correlation two nodes have.

3.4.3.2 Recommendation When the job comes to a recommendation, we have two cases. I assumed we are wanted to give five recommendations end of the day.

The first one is a single widget usage. When the user uses a single widget, I reached to that widget node and gave the five widgets a recommendation in which nodes have the highest five edges with the used widget. For an example from the visual graph, when someone used widget 6, it is reasonable to recommend widget five and widget 4 to that person, but there is no logic to recommend widget 8.

The second case is multiple widget usage. In this situation, things get a little bit complicated because there are different widgets usages, and there are different edges. Different ways could be followed, but I chose to add all the edges of all nodes and treat them as one node to approach this case like in the first case. For another example from the visual graph, I said there is no logic to recommend widget 8 to someone using widget 6. There can easily be a case that someone uses widget 6 and widget 5 together since there is a correlation between them. If someone suddenly uses both of them, widget 8 becomes a perfect choice to offer to the user.

3.4.4 Implementation

I was aware there were different graph libraries available, but I chose to write my graphs using python dictionaries. The main graph was the leading dictionary that contains widgets as keys of the dictionary, and nodes were also dictionaries that hold edge values. Node dictionaries hold other nodes as keys and values as the edge values for every individual node. Here is a shortened node dictionary example:

"Birth Date Picker": { "Bootstrap Switch Field": 675,
"Cities of Indonesia": 47,
"E-mail Validator": 851,
"Field Multiplier": 1627,
"Form Tabs": 1563,
"Global Countdown": 200,
"Image Picker": 835,
"Mobile Responsive": 1417,
"Phone Picker": 747,
"Progress Bar": 1323,
"Show Map Location": 381,
"Smooth Signature": 2375,
"Take Photo": 1810,
"Terms Conditions": 3602,
"YouTube": 611,
"E-Signature": 5509,
"Auto Complete": 1017,
"Barcode Scanner": 304,
"Checkbox in Dropdown": 1249,
"Code Editor": 370,
"Day Countdown": 422,
"Digital Clock": 184,
"Disqus": 30,
"Double the Donation": 64,
"Draw on Image": 408,
"Email Correctness Check": 1352,
"Facebook Comments": 175,
"Fancy Radios": 194,
"Fancy Timer": 274,
"Get Form Page URL": 775,
"Gift Registry": 316,
"Kinomap": 187,
"Like and Dislike": 155,
"Livestream": 39,
"Mini Date Picker": 964,
"Minimal Radio Buttons": 160, }

"Birth Date Picker" represents the main widget; keys represent the node names, and values are the edges' values with the main widget. That value means the widget "Birth Date Picker" used together with a widget {value} times.

3.4.4.1 Code I want to share how I wrote the code and which functions I used for graph implementation. I used separate functions for this solution. After I finished writing this algorithm, I offered to write it again using python class, but the mentor said there is no need to do that.
Here are my functions and their purposes:

init_graph()

Returns an empty dictionary

---

add_node(graph,node)

INPUTS:
Graph: type=dict base dictionary for the graph
Node : type=str target widget to add

METHOD:
Adds target widget to graph and add edges to all other nodes of
target node

---

init_edges(graph,node)

INPUTS:
graph: type=dict base dictionary for the graph
node : type=str target node whose edges to be added

METHOD:
Adds all of the other nodes as edges to target node

---

parse_cols(string,sep=',')

INPUTS:
string = str to be parsed
sep = target seperator for string

RETURNS:
Splitted version of string

---

edge_up(graph,node,target)

INPUTS:
graph: type=dict base dictionary for the graph
node : type=str base node edges will be added
target: type=str target node will be added to base node as edge

METHOD:
It increases the edge between the node and target node by one.

---

base_integrate(base,target)

INPUTS:
base : type=dict edges of a node
target: type=dict edges of the target node

METHOD:
Integrates the edge values of two node

---

recommend(graph,widgets)

INPUTS:
graph : type=dict base dictionary for the graph

widgets: type=list widgets that user has used

METHOD:
Integrates the edges of all widgets and recommend 5 widgets which have the most powerful edges

---

init_save_graph(filename)

METHOD:
Inits the graph,nodes,edges according to the data. Prints it as json string

---

load_graph(filename)

Returns saved graph

3.4.5 Results

When I took the solution to my mentor, he found the solution very sensible and friendly. He told me to test the algorithm look if it gives sensible recommendations or not. Here is my testing method; I am getting a widget from every line and recommending it. If one of the results is in the same line with the main widget, I count this recommendation successfully, otherwise failed. It is reasonable to look if one of the recommendations is in the line because it shows we offered a widget to the user that can be used together. Results are like this:

By recommending 5 widgets: 77.80%

By recommending 3 widgets: 69.30%

This means 77.80 percent of my recommendations were in the same line with the used widget.

After this result, the mentor finds this is a successful algorithm, thanked me.

3.5 Fifth Week

3.5.1 New Project

By the fifth week, which is the last week of the internship. I take my third and last project. I asked my mentor to give me a project instead of choosing myself. He gave me the fifteenth project of the project list. Which is 'Detect Must-Have Questions'. This project aims to recommend must have questions to the user for specific form types. The desired algorithm should be trained with the dataset and recommend questions by checking the user's previously used question. For Example, a user wants to prepare an order form and adds some t-shirts and prices to the form. Our recommendation system should recommend the payment question type. In general, this project is very similar to the previous project, which is why the mentor gave me this project.

3.5.2 Dataset Inspection

The dataset consists of 100 000 lines of inputs. These lines contain 1 column that is formed of questions separated by commas. Every line represents a form, and question types in the line the types used by the user for that form. Dataset is also very similar to the previous one. Here are some examples of the dataset:
0 control_autoincrement,control_divider,control...
1 control_address,control_autoincrement,control...
2 control_address,control_datetime,control_email...
3 control_address,control_birthdate,control_date...

4 control_email,control_fullname,control_payment...
... ...
99995 control_checkbox,control_datetime,control_emai...
99996 control_checkbox,control_dropdown,control_text...
99997 control_checkbox,control_datetime,control_drop...
99998 control_checkbox,control_datetime,control_full...
99999 control_divider,control_dropdown,control_email...

3.5.3 Idea&Solution

When I inspected the project naturally, I thought I could use the algorithm I wrote for widget recommendation. After some extra examination, I decided to use it. The graph-based solution but instead of widgets as nodes, there were question types as nodes. There was a slight difference between the two projects: I had nearly 300 widgets in a previous project, but in this one, I nearly had 100 types.

3.5.4 Implementation

For the implementation, I used the same code for this project and changed only the functions' comments since there were no widgets in this project. Here are the functions and comments:

```
    init_graph()

Returns an empty dictionary
```

```
add_node(graph,node)

INPUTS:
Graph: type=dict base dictionary for the graph
Node : type=str target types to add

METHOD:
Adds target type to graph and add edges to all other nodes of
target node
```

```
init_edges(graph,node)

INPUTS:
graph: type=dict base dictionary for the graph
node : type=str target node whose edges to be added

METHOD:
Adds all of the other nodes as edges to target node
```

```
parse_cols(string,sep=',')

INPUTS:
string = str to be parsed
sep = target seperator for string

RETURNS:
Spliced version of string
```

```
edge_up(graph,node,target)
```

INPUTS:
graph: type=dict base dictionary for the graph
node : type=str base node edges will be added
target: type=str target node will be added to base node as edge

METHOD:
It increases the edge between the node and target node by one.

---

base_integrate(base,target)

INPUTS:
base : type=dict edges of a node
target: type=dict edges of the target node

METHOD:
Integrates the edge values of two node

---

recommend(graph,types)

INPUTS:
graph : type=dict base dictionary for the graph
types: type=list types that user has used

METHOD:
Integrates the edges of all types and recommend 5 types which have the most powerful edges

---

init_save_graph(filename)

METHOD:
Inits the graph,nodes,edges according to the data. Prints it as json string

---

load_graph(filename)

Returns saved graph

3.5.5 Results

When I told my mentor that I used the same algorithm for the project, he welcomed this solution and accepted it. I wanted me to test it like the previous one. I test the algorithm with the same method. Here are the results:

By recommending 5 types: 97.65%

By recommending 3 types: 95.12%

By recommending 1 types: 76.59%

These results are way higher than the previous one because the node number is less than the widget project, and also the randomness of the data is way lesser than the widget project. I checked that if there is an overfit or something, algorithms were trained with 80% of the data. I used the 20% of data to test the algorithms.

4 Conclusion

I completed my internship at JotForm remotely. After completing I went to the office to visit. They have a really nice and motivating office I wish to do my internship face to face, but due to do pandemic, there is nothing to do. Remote internship was also not a bad experience at all. In fact, it was a remarkable experience.

I worked on 3 projects and all of them developed me in different ways I can say. I've learned a brand new library for me that is TensorFlow. I've learned so much about Generative Adversarial Networks which is an amazing network that impressed me. I've learned so much about Graphs, even I've not taken the Data Structures course. Finally, I've learned so much about mechanism of a market company. It was nice to do my internship full time regarding this.