Bilkent University

Department of Computer Engineering

CS 492: Senior Design Project II

Spring 2022

Final Report

Project Name: *Laber*

Group Members:

Yiğit Gürses	21702746	yigit.gurses@ug.bilkent.edu.tr
Melisa Taşpınar	21803668	melisa.taspinar@ug.bilkent.edu.tr
Onur Oruç	21702381	onur.oruc@ug.bilkent.edu.tr
Emin Adem Buran	21703279	adem.buran@ug.bilkent.edu.tr
Mustafa Hakan Kara	21703317	m.kara@ug.bilkent.edu.tr

Supervisor: Assoc. Prof. Can Alkan

Innovation Expert: Ahmet Eren Başak

Jury Members: Assoc. Prof. Can Alkan

Asst. Prof. Dr. Shervin Arashloo

Asst. Prof. Dr. Hamdi Dibeklioğlu

Project Webpage: https://eminademburan.github.io/laber/

Table of Contents

1. Introduction

With the coming of the information age and rise of social media, there is an increasing quantity of information available on the internet that can be taken advantage of [1]. Companies try to take advantage of this using natural language processing and human analysts but they both come with some disadvantages. Machine learning models try to approximate human expertise and always come with a margin of error [2], while processing a large amount of data with human experts can be very costly. In order to reduce this cost, crowdsourcing platforms such as Amazon's Mturk [3] can be used but such unspecialized platforms require expertise and investment on the side of the client. For these reasons, there exists a niche to be filled by specialized platforms that are easy to use for both the clients and human experts.

Laber is a mobile based platform that aims to provide real time social media analysis by human experts. Clients are able to crowdsource their work with minimal knowledge of the system and have access to a pool of human experts at all times. The obtained analytics are available to the clients through our website. The experts are able to do all of their work through a mobile application allowing them to work remotely and efficiently. Various features are implemented to ensure the reliability of our experts and provide them the optimal environment to do their work.

2. Requirements Details

2.1. Final Functional Requirements

Users can sign up as a Client or an Expert. Clients can register through a web browser. Experts can register through the mobile application with an SMS confirmation.

2.1.1. Task Creation

The Clients are able to create task streams using the website. They;

- Can specify keywords and hashtags to be searched for in Twitter.

- Can specify a set of metrics and their intervals/categories. These metrics are highly customized so that the Clients can enter any metric of their choice. For example: {sentiment: [-1, 1], sarcasm: (serious, sarcastic, mixed)}.
- Can specify a time interval. The beginning and the end of the interval could be any two date-times. The end of the interval can also be indefinite, which means new posts will keep getting scraped until the task stream is stopped manually.

2.1.2. Task Analysis

The Customers are able to view the results of created tasks that have been evaluated by the Experts.

- For scalar task results, the score is displayed based on the minimum and maximum score that are specified while the Customer is creating a task.
- For nonscalar task results, the results are displayed on both a bar chart and a pie chart.
    - For each metric, the scores to metric options are displayed on the bar chart. For example if the metric is sarcasm and if the metric options are serious, sarcastic, mixed and if the results to the metric options are 3, 10, 4, respectively, they are all displayed on the bar chart.
    - The results are also displayed on the pie chart to make it easier for the Customer to visualize the score distribution among all metric options.

2.1.3. Evaluate Task

The Experts are able to evaluate tasks using the mobile application based on the metrics displayed to them.

The system distributes the samples across the currently active Experts.

- The same sample is given to multiple Experts. Ideally, to a mix of highly reliable Experts and newly registered ones so that the new ones can be evaluated based on the comparison of their results.

The Experts will evaluate the samples based on the given metrics.

- The sample is shown in full context and all work will be done through the mobile app.
- Experts gain points for each task completed based on their reliability score. Experts with higher reliability scores will earn more points.

2.1.4. Voice Chat

When there is a high variance in the results of a task, the active Experts that evaluated the task previously are asked to join a voice chat to discuss the task and reach a consensus.

2.2. Final Non-Functional Requirements

2.2.1. Usability

- Clients can create tasks and see the analytics through a browser without any technical knowledge.
- Experts can do all of their work through our mobile application without any technical knowledge.
- Experts should be able to communicate with each other well using voice chats. To maintain the quality of the audio communication as well as the usability of the voice chat, there are not any large delays or loss of data.
- The application is efficient and runs smoothly to not interrupt the workflow of experts.

2.2.2. Availability

Laber must be available 7/24 because the scrapper algorithm fetches the tweets with the corresponding keywords and hashtags continuously and distributes the tasks to the Experts. Another important reason for availability is that the Experts should be able to evaluate the tasks assigned to them any time any time during the

day. Therefore, backend services have been deployed to AWS as its reliability is proven over time.

2.2.3. Portability

The system supports various mobile devices and web browsers.

3. Architecture, Design and Implementation Details

3.1. Packages

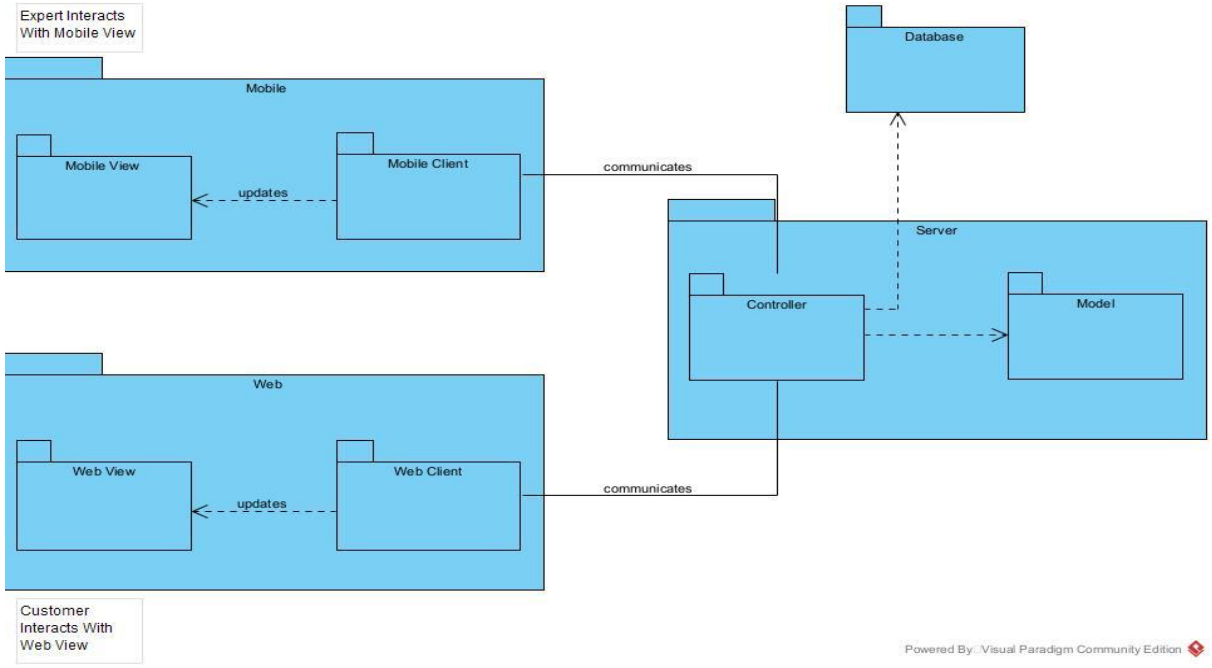

Figure 1: Packages

A basic overview of the subsystems include the mobile client where experts receive tasks and give their answers, a web client where customers create new projects and look at the data obtained from their previous projects, and finally, the server side code where controller classes handle requests from these clients and model classes encapsulate the necessary persistent data.

3.1.1. Clients

3.1.1.1. MobileClient

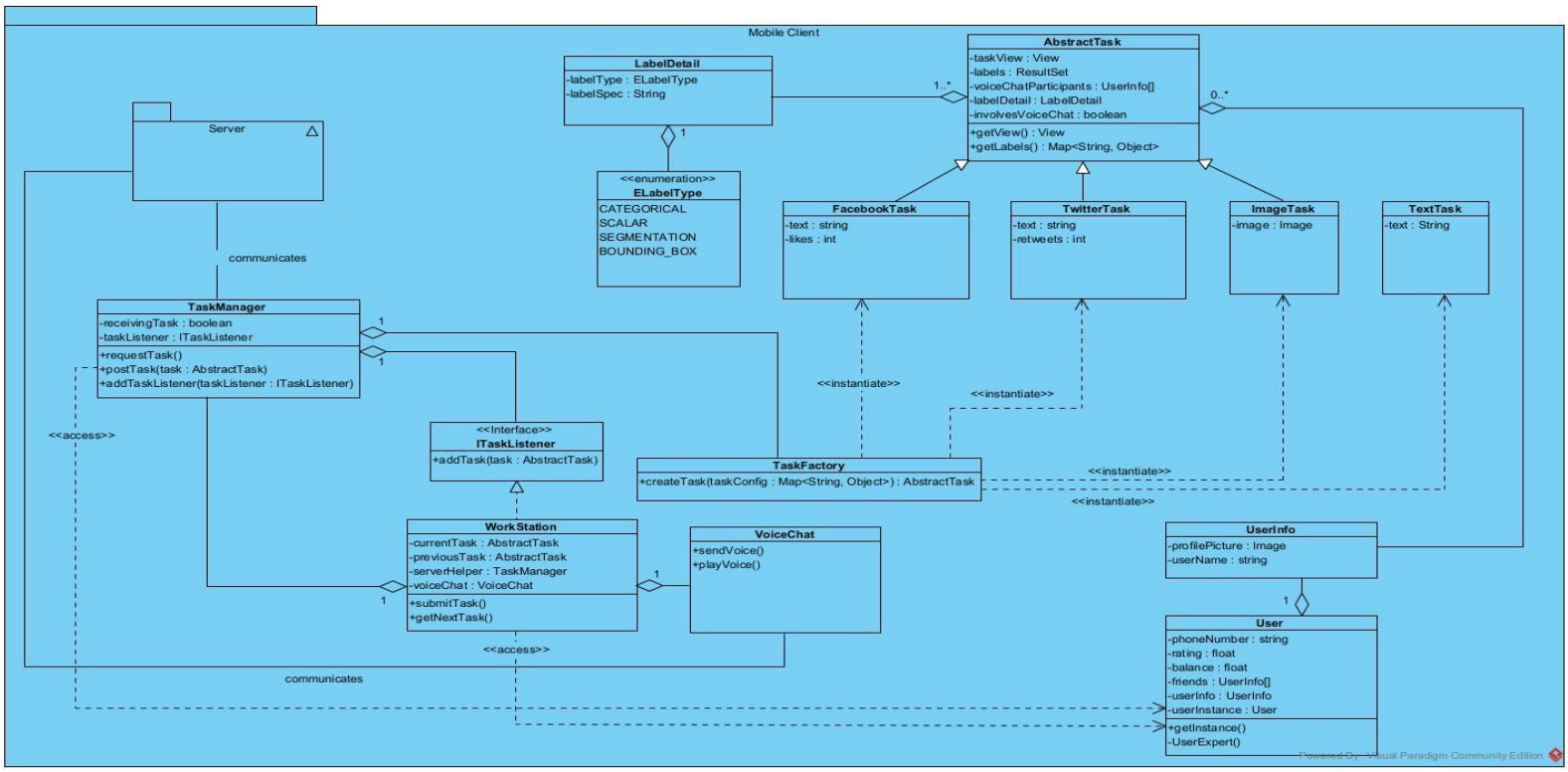

Figure 2: Mobile Client

Views:

ExpertProfileView: It will show details of the user account such as name, balance, friends, etc.

ExpertSettingsview: It will allow experts to update profile information and change the application settings such as audio, dark theme, etc.

TaskView: It displays the tasks (e.g. Tweet, Facebook post) to be evaluated to the experts.

PaymentView: It displays the payment options to the Expert to withdraw his/her funds from the application to his/her either bank account or cryptocurrency wallet.

FriendsView: It displays the list of friends of the Expert.

ForumView: It displays the forum in which the Experts discuss the options of a specific task to come up with an accurate evaluation.

VoiceChatView: It displays the voice chat page in which the Experts discuss a pre-evaluated task that has a high variance to come up with a better evaluation.

Models:

AbstractTask: It is responsible for creating a variety of tasks such as Facebook tasks and image tasks. It also manages information about a task regarding voice chat and discussion forum.

FacebookTask: It manages the information about Facebook tasks like the text in the post and the number of likes it received.

TwitterTask: It manages the information about Twitter tasks like the text in the tweet and the number of likes/retweets it received.

ImageTask: It manages the image related data.

TextTask: It manages plain text data.

Controllers:

TaskManager: It requests tasks from the database and conveys them to the workstation. It also takes the answers of Experts bundled by the WorkStation and pushes them to the Server.

WorkStation: Handles updating the views based on the tasks received from the database. Takes the answers from the Expert and sends them to the TaskManager when they are ready to be pushed.

3.1.1.2. WebClient

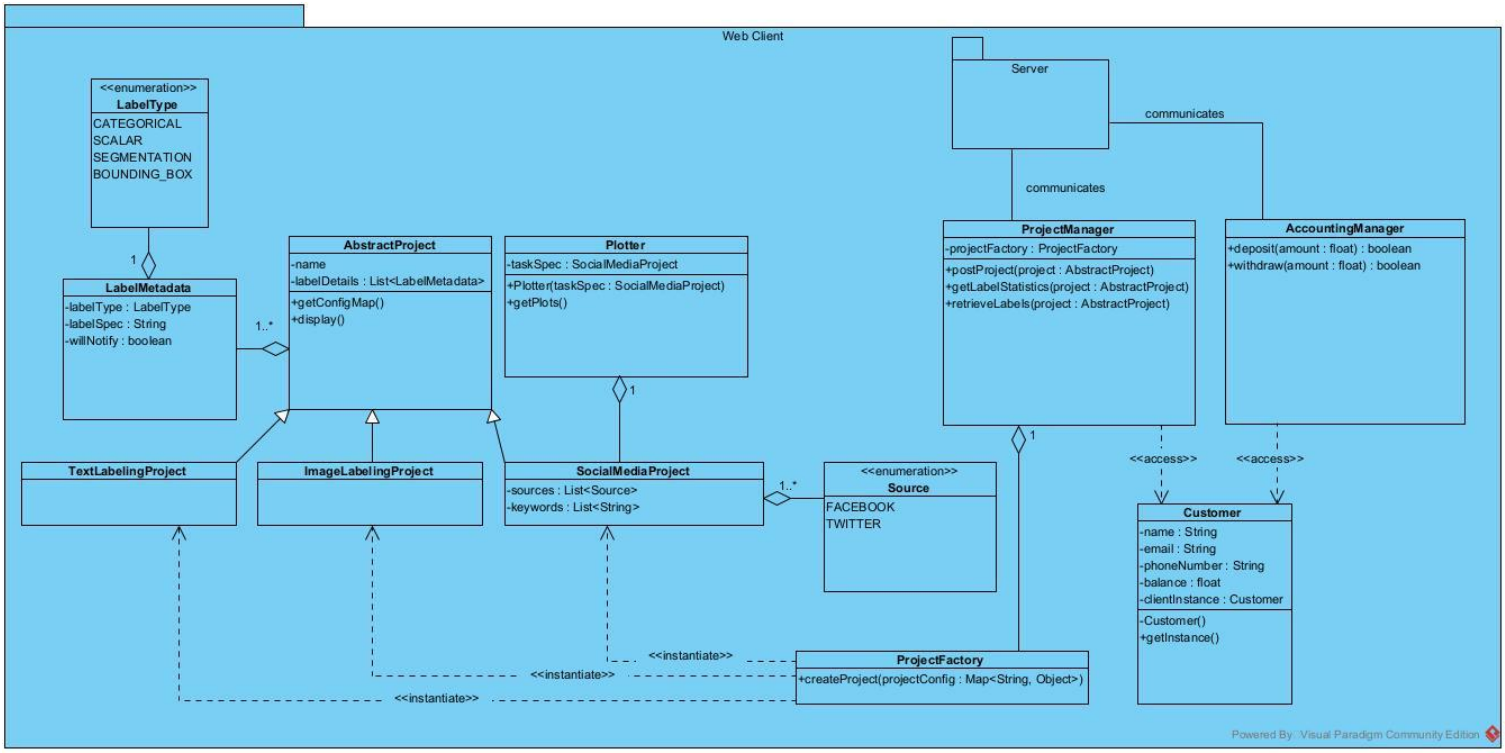

Figure 3: Web Client

Views:

TaskCreationView: It displays the options for a task to be created such as gender of Experts, their ages, ethnicity, etc.

AnalysisView: It displays the analysis of evaluated tasks to the Customer.

Models:

Plotter: It receives the specifications and statistics related to a social media project and displays the plots for task analysis.

AbstractProject: It manages the configuration related problems while creating a task.

Controllers:

ProjectManager: It posts projects to the server and receives statistics related to a task from the server.

AccountingManager: It communicates with the server for balance related operations which are deposits and withdrawals.

3.1.2. Server

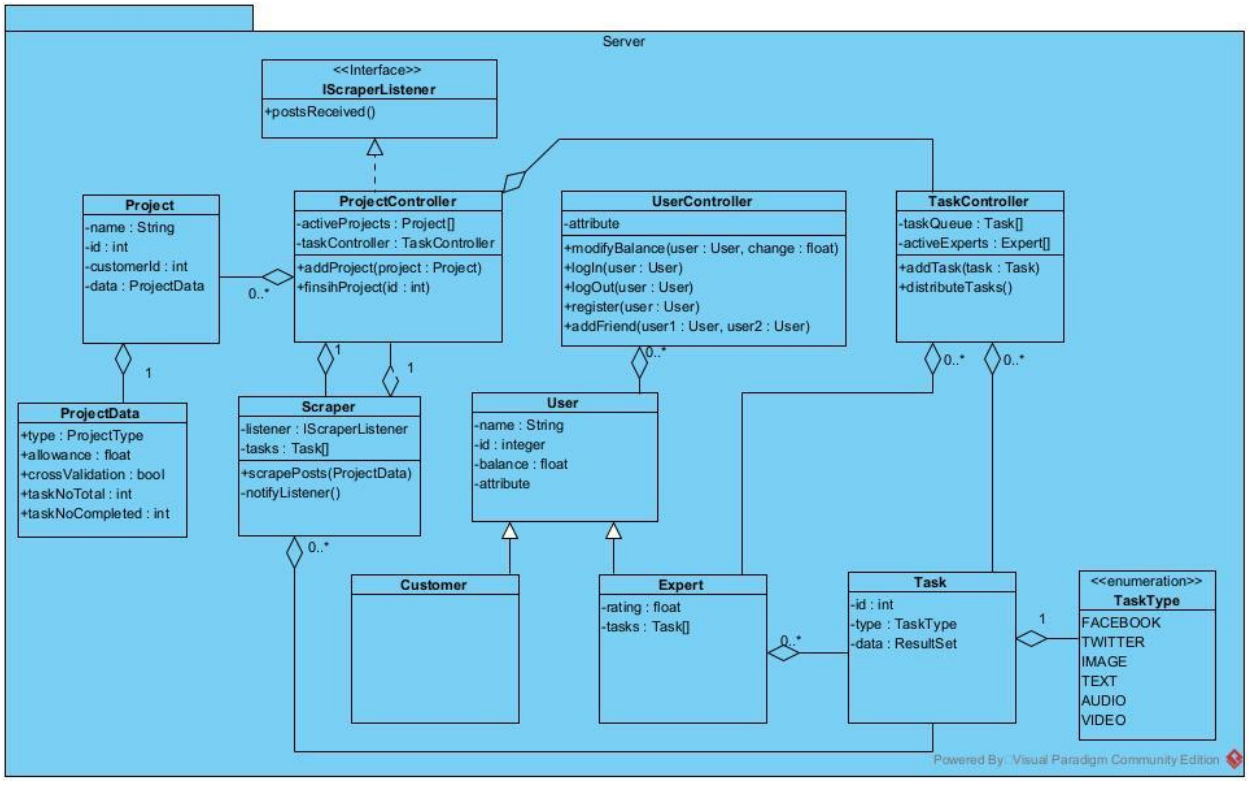

Figure 4: Server

Controllers:

ProjectController: It manages the active projects, uses the class Scraper to scrape new tasks from the internet if the Project requires it. Generates the new tasks based on active projects and gives them to the TaskController. If the Project is completed, it is pushed to the database and removed from the active Projects list.

UserController: It manages a set of operations such as login, registration, payment, task evaluation, etc.

TaskController: It manages task related operations such as choosing the tasks to be displayed to a specific Expert, task distribution, defining the reward of each task, etc.

Models:

Customer: Customers are responsible for creating tasks, uploading datasets to be evaluated, choosing the types of Experts they want to work with, specifying the platform of their choice (i.e., Facebook or Twitter). They also have access to the analysis of the evaluation results which will be displayed on the website. The Customer model in the server architecture encapsulates this information.

Expert: Expert models include the data on Experts' user information along with the ids of tasks they completed. The server will process these tasks and give the Experts ratings based on their performances obtained from the cross validation process.

Task: This class encapsulates all data related to a task that will be given to an Expert. Tasks awaiting to be completed will be inserted to a TaskQueue in the server consisting of task models. The TaskController will handle the distribution of the tasks in the Queue to the active Experts.

Project: This class encapsulates all data related to a Project that can be created by a Customer from the Web Client. It includes the project name, owner customer's id, project type (dataset labeling /social media analysis), maximum allowance etc.

3.2. Class Interfaces

3.2.1. Mobile Client Classes

<<class>> TaskManager:	
Retrieves new tasks from the server and posts their results.	
Attributes	
-receivingTask : boolean -taskListener : ITaskListener	
Methods	
+requestTask() +postTask(task : AbstractTask) +addTaskListener(taskListener : ITaskListener)	Sends a task request to the server. Posts the result of a completed task. Registers a new task listener.

<<interface>> ITaskListener:	
TaskManager sends new tasks through this interface.	
Methods	
+addTask(task : AbstractTask)	A callback for task adding.

<<class>> WorkStation:	
Provides common attributes for all the concrete task classes.	
Attributes	
-currentTask : AbstractTask -previousTask : AbstractTask -serverHelper : TaskManager -voiceChat : VoiceChat	
Methods	
+submitTask() +getNextTask()	Triggered when the expert submits a task. Requests a new task from TaskManager.

<<class>> VoiceChat:
Provides common attributes for all the concrete task classes.

<<class>> User:	
The model class of the expert.	
Attributes	
-phoneNumber : string -rating : float -balance : float -friends :UserInfo[] -userInfo : UserInfo -userInstance : User	
Methods	
+getInstance() -User()	Returns the unique instance of the class. Private constructor for the Singleton Design Pattern

<<class>> UserInfo:
The model class of the expert.
Attributes
-profilePicture : Image -userName : string

<<abstract class>> AbstractTask:
Provides common attributes for all the concrete task classes.
Attributes
-taskView: View -labels: ResultSet -voiceChatParticipants: UserInfo[] -labelDetail: LabelDetail -involvesVoiceChat: boolean

<<class>> FacebookTask:
Contains specific details for the task involving Facebook posts.
Attributes
-text : string -likes : int

<<class>> TwitterTask:
A concrete task class for the Tweet analysis tasks.
Attributes
-text : string -retweets : int

<<class>> ImageTask:
A concrete task class for image labeling.
Attributes
-image : Image

<<class>> TextTask:
A concrete task class for plain text labeling.
Attributes
-text : String

<<class>> TaskFactory:
Generates different types of concrete task instances.
Methods

+createTask(taskConfig : Map<String, Object>) : AbstractTask

3.2.2. Web Client Classes

<<class>> ProjectManager:	
Handles project generation and data retrieval.	
Attributes	
-projectFactory : ProjectFactory	
Methods	
+postProject(project : AbstractProject) +getLabelStatistics(project : AbstractProject) +retrieveLabels(project : AbstractProject)	Posts the project to the server. Generate specified statistics out of labels. Retrieves labels from the database.

<<class>> AccountingManager:	
Performs the monetary transactions of the customer.	
Methods	
+deposit(amount : float) : boolean +withdraw(amount : float) : boolean)	Increases the credit of the customer. Makes a refund to the customer.

<<class>> Customer:	
The model class of the customer.	
Attributes	
-name : String -email : String -phoneNumber : String -balance : float -customerInstance : Customer	
Methods	
-Customer()	Private constructor to apply Singleton Design

+getInstance()	Pattern. Returns the unique instance for the customer.

<<class>> ProjectFactory:	
Generates different types of concrete projects.	
Methods	
+createProject(projectConfig : Map<String, Object>)	Creates a new project instance based on the given project config.

<<class>> AbstractProject:	
Implements the common features of concrete projects.	
Attributes	
-name : String -labelDetails : List<LabelMetadata>	
Methods	
+display()	Displays the project details.

<<class>> TextLabelingProject:
A concrete project class for plain text analysis.

<<class>> ImageLabelingProject:
A concrete model class for image labeling projects.

<<class>> SocialMediaProject:

A concrete model class for social media analysis projects.
Attributes
-sources : List<Source> -keywords : List<String>

<<class>> Plotter:
Displays the results of data analyses as plots.
Attributes
-taskSpec : SocialMediaProject

<<data class>> LabelMetadata:
Holds information about a label specification.
Attributes
-labelType : LabelType -labelSpec : String -willNotify : boolean

3.2.3. Server Classes

<<class>> ProjectController:
Handles the addition of new projects to the database. Handles the task creation process for active projects. The created tasks are passed to the TaskController class to be then distributed to active Experts when possible.
Attributes
-activeProjects : Project[] -taskController : TaskController

Methods	
+addProject(project : Project) +finishProject(id : int)	Adds a newly created project received from a web client to the active Projects list. When all tasks of a project is given to the TaskController, it is removed from active Projects.

<<class>> Project:
Encapsulates the data related to a Project.
Attributes
-name : String -id : int -customerId : int -data : ProjectData

<<class>> UserController:	
Handles the registration, login, logout, balance changes and friend requests for users.	
Attributes	
-user : User[] -taskController : TaskController	
Methods	
+modifyBalance(user : User, change : float) +logIn(user : User) +logOut(user : User) +register(user : User) +addFriend(user1 : User, user2 : User)	Changes balance of given user by given amount. Tries to log in a given user. Tries to log out a given user. Register a given user as a new user. Modifies the database to recognize the given users as friends.

<<class>> TaskController:

Receives newly created Tasks from the ProjectController. Distributes the tasks to active Experts.	
Attributes	
-taskQueue : Task[] -activeExperts : Expert[]	
Methods	
+addTask(task :Task) +distributeTasks()	Adds the given task to the Task Queue which contains Tasks awaiting to be distributed to an expert. Distributes a number of tasks from the Task Queue to the available Experts. This number will change depending on the number of active Experts.

4. Testing

To test the application, we mostly used the pair programming technique to detect bugs quickly. The codes were reviewed by other group members. The group members used the application and tested the reaction of the application with extreme inputs to test edge cases.

5. Maintenance Plan and Details

5.1. Database Maintenance

Laber uses Mongo DB as the database. The task details specified by the Customer, the results of the evaluated tasks, the task evaluations of Experts, and many other important things are stored in the database. Therefore, keeping the database updated continuously and synchronized is crucial for *Laber.* To make maintenance easier in the future, the best practices were followed using the Mongo DB documentation to create an efficient, clean, and robust database.

5.2. Server Maintenance

Laber has been deployed to AWS. To keep the server up and running, updates made to the AWS should be regularly followed and make necessary adjustments. To increase efficiency and scalability, developers should continuously optimize the server as the number of users increases.

6. Other Project Elements

6.1. Consideration of Various Factors in Engineering Design

During the analysis phase of our project, we have had to consider several factors relevant to engineering design.

- Public Health: Our approach allows people to work as much as they want when they want. These, at first glance, might sound like positive qualities and they can be. However, they can also have downsides. Extensive use of our application can disrupt a user's work life balance and can be detrimental to their mental and physical health. For this reason, in order to encourage a regular use of our application, we will be providing bonus payment for a predetermined amount of work each day. Excessive work done after the user surpasses this predetermined amount will not yield any bonus payment. We hope that this will discourage users from working in huge chunks of time. We might also consider sending the users messages that recommend taking a break after working for a long period.
- Public Safety: People that work on our application are also likely to work on other platforms that require frequent car-driving such as Uber or DoorDash. Along with that, many of our users will be driving a car daily to go to the places they need. If one of our users decide to work on our application while driving a car at the same time, this will create a safety concern for both the user and the pedestrians and drivers in the area. For this reason, we thought of using GPS to limit the user's maximum speed to be eligible to work on our application. However, with that approach, we prohibit users on buses and

trains to work and that would be an unwanted consequence. Instead we decided to make the users sign an agreement form that requires them to not use our application while driving a car or doing any other dangerous activity that requires uninterrupted focus.

- Public Welfare: Our application can provide work opportunities to people, only requiring a mobile phone and internet access as an investment. Therefore, it can be used by people in need as an easy income source. However, there are a couple of considerations regarding this point. Firstly, some people may have access to only a very low end mobile phone with an older operating system. We need to make sure our application can run on such devices if we want to allow these people to work on our application. For this reason, we need to write efficient code that does not use unnecessary resources. The application should also not be heavy on the battery usage. Secondly, the application will need to exchange data with our servers very frequently. Some users will be paying their internet providers based on the amount of data they download and upload. If our application exchanges any unnecessary data, it might be unprofitable to work on our application for many users. For this reason, we will try to minimize the data exchange via compressing data and not sending/receiving any unnecessary information.
- Global: We plan to allow any expert to work on any project they are eligible for. This means, an expert might be working on a project created by a client from a different country. In this case, different regulations from different regions will need to be taken into consideration.
- Cultural: Since the tasks given to the experts will be very generic, there were not any cultural factors taken into consideration when designing the project.
- Social: During their work sessions, Experts might have to interact with other Experts through our voice chat feature. This might cause problems if some mal-intentioned Experts start harassing others. In order to minimize this risk, we will implement a report functionality and also bans on phone numbers. Since the application requires an SMS confirmation for registration, a ban on a phone number should be able to reliably prevent banned users from using our application.
- Environmental: It is expected that our program will run in many different devices and might use a cryptocurrency for some of the transactions. For energy conservation, it is essential that the program runs efficiently in terms of battery usage, and the selected

cryptocurrency is also environmentally friendly. It should be noted that our application will allow Experts to work remotely and this will reduce their carbon footprint by reducing their transportation needs.

- Economic: It is expected that our application will require a big number of small transactions. These transactions may be from across the border which may increase their cost. In order to reduce the number of transactions, we will set a minimum amount that can be withdrawn from an Expert's account. Also, alternative to institutions that provide transaction services like banks, we will provide an option to get paid in a cryptocurrency that our platform supports. In the case that this feature gets implemented in the final product, it will be very important to select a low fee currency that does not have a big environmental impact.

	Effect Level	Effect
Public Health	6	Daily bonuses for a limited amount of work, Warning messages for prolonged work sessions
Public Safety	7	Safe usage agreement form upon registration, Warning message when high speeds detected while using our application
Public Welfare	7	Program needs to run on low end devices, Low battery usage, Low internet usage through data compression etc.
Global Factors	3	Different regulations need to be taken into consideration for different regional versions of the application
Social Factors	7	Report functionality, SMS confirmation on registration and ban on phone numbers
Environmental Factors	8	Efficient code, Environmentally friendly transaction alternatives
Economic Factors	8	Limit on minimum transaction amount, Low fee transaction alternatives

Table 1: Factors that can affect analysis and design

6.2. Ethics and Professional Responsibilities

Our project includes processing data. As the data may be sensitive, one of our ethical issues is to protect the data and ensure data privacy so that our clients may feel secure while using Laber. To ensure data privacy, we do not use the data without users' details.

Malicious experts may abuse the labeling process by making an agreement to label the post incorrectly in a way that results in significant errors in the task. Our responsibility is to force requirements to prevent this from happening. One of the precautions to prevent this is to validate the experts' expertise level. Moreover, occasionally, the experts are asked to label posts that we already have accurate labels related to. If they mislabel a significant amount of these posts, their score is reduced severely and they will not be able to label posts after some reduction in their scores.

6.3. Judgements and Impacts to Various Contexts

Throughout the design and implementation of our project, we have made some judgments that had impacts in global, economic, environmental and societal contexts.

Judgment Description	Laber must be up and running 7/24 to distribute tasks to the Experts continuously.	
Context	Impact Level	Impact Description
Global Impact	10/10	Both Customers and Experts should be able to access Laber anywhere in the world at any time.
Economic Impact	9/10	Laber will offer job opportunities to many Experts since Experts are able to earn money after evaluating tasks
Environmental Impact	9/10	To use Laber, the environmental requirements are quite low. Experts will be able to use Laber anywhere as long as they have a mobile phone and internet access.
Societal Impact	10/10	Laber will offer job opportunities to many Experts since Experts are able to earn money after evaluating tasks. This way, people in the society will have a chance to be employed.

Table 2: Availability judgment

6.4. Teamwork Details

6.4.1. Contributing and functioning effectively on the team

We have tried our best to be not only a functional team, but also a collaborative and inclusive one. We can say that each member has contributed and functioned effectively in our project. To be more precise, we followed the distribution of the work packages among our team that we had provided in our project-plan.

WP#	Work Package Title	Leader	Members Involved
WP1	Mobile App	Hakan	Melisa, Yiğit
WP2	Voice Chat	Emin Adem	Yiğit, Hakan
WP3	Server & Distribution System	Yiğit	Melisa, Hakan, Emin Adem
WP4	Database	Yiğit	Hakan, Emin Adem, Onur
WP5	Website	Onur	Onur, Hakan
WP6	Reports	Melisa	Onur, Emin Adem

Table 3: Work packages

6.4.2. Helping creating a collaborative and inclusive environment

We share responsibilities. When someone responsible for a task has other responsibilities or s/he finds the task given to him/her hard to complete, we help him/her finish the task. When someone is confused about a task, we always help each other to clarify what should be done exactly. We periodically inform each other about the status of the tasks we are responsible for to keep each other up to date. Even if we assign different tasks to each other, we always ask other teammates' opinions while we are working on a task or after finishing the task to ensure that the tasks have been finished in the correct way. We give each other feedback for the tasks we are responsible for and make changes accordingly.

6.4.3. Taking lead role and sharing leadership on the team

We always shared the leadership responsibility. When we needed to plan a meeting, determine our schedule, or do something together in general, whoever was the least busy at the moment took responsibility to lead others. The reason behind this is for both to reduce each other's burden and add different perspectives to the way we proceed. Moreover, even if we have different leaders at different times, it does not mean that the leader makes choices on his/her own. The responsibility of our leader is to organize people, combine different thoughts and come up with a plan by putting the thoughts together.

6.4.4. Meeting objectives

Originally, in our project-plan, the milestones we planned to hit were as below.

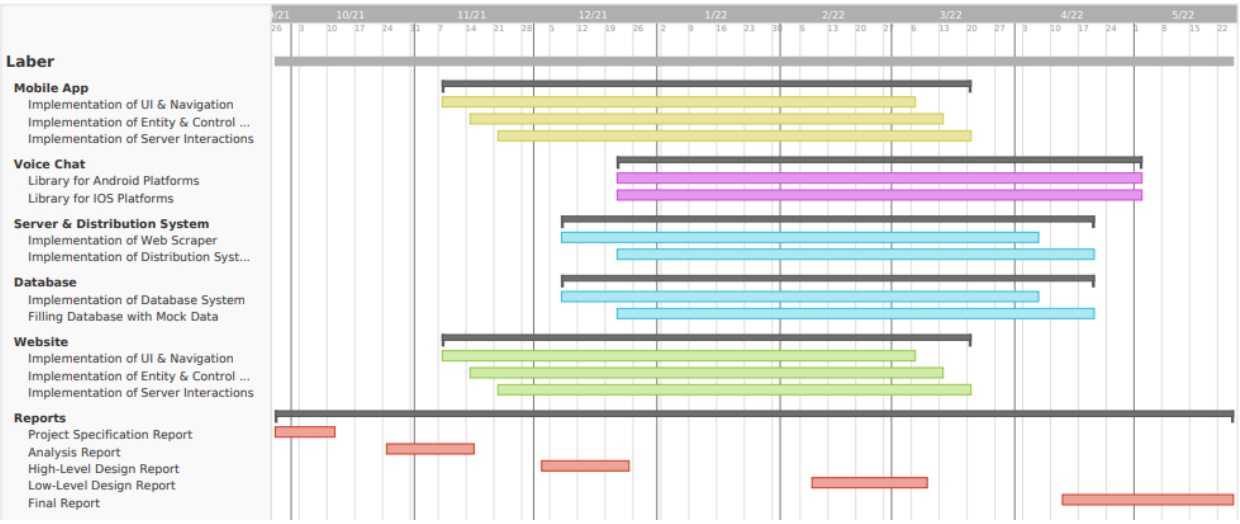

Figure 5: Milestones in our project-plan

Among these milestones,

- For the mobile app, we have completed the implementations of UI & navigation, as well as server interactions; but not entity & control (model & control in MVC).
- We have completed the tasks specified under the voice chat title.
- On the topic of server & distribution system, we have implemented both the web scraper and the distribution system.
- As for the database, we have completed the implementation of our database system. However we have not filled the database with mock data.
- For the website, we have completed the implementations of UI & navigation, as well as server interactions; but not entity & control.
- We have completed and submitted all of our reports in time.

6.5. New Knowledge Acquired and Applied

While analyzing our project, we have also considered what new knowledge we may need to realize it and how we can acquire that knowledge. In order to make our project a valuable learning experience, we have tried to venture into topics and make use of tools that we are not already familiar with. In other words, turning our project into an opportunity for learning was itself one of our considerations during the analysis phase.

For instance, as mentioned before, we have decided to use the React Native framework for our mobile application. This allowed us to create an application that can be available on multiple platforms, while also making use of React's rich libraries. However, it was also a great learning experience for all members as none of us has had any substantial experience using React Native. We acquired the knowledge we need on this topic with online learning and some hands-on experience, i.e. learning by doing.

Furthermore, our project includes a server side, with which we were all a bit unfamiliar. In order to acquire the knowledge needed on this topic, we made use of several learning strategies such as online learning using documentations and tutorials, learning from peers that are knowledgeable on the subject, and of course learning by doing.

All in all, our project was an ideal learning experience as it had several distinct sub-tasks and parts such as a mobile application, a website, a database, a server side, etc. To get the knowledge we have needed throughout our project, we used online resources, learning from each other, learning by experimenting, getting help by people knowledgeable in the field; noting that these corresponded to learning strategies such as online learning, learning from peers, learning by doing and interviewing experts.

7. Conclusion and Future Work

All in all, we believe we have succeeded in our goal to make *Laber* a project that is easy to extend and maintain. This should help us greatly with the future work that needs to be done. Still, we should complete the milestones we had specified in our project-plan but did not have the time or resources to complete. Furthermore, we plan on improving the project based on the feedback we get from the users, as well as any new ideas that may come to our minds. Such improvements can consist not only of changing some existing features, but also adding new ones.

We have found the process of designing and implementing *Laber* not only enjoyable, but also challenging. However, we have learned to get over the obstacles we have faced, and to function as a team. We had each other's backs when needed, which resulted in a collaborative and inclusive atmosphere. Furthermore, this project not only helped us learn a lot of new things on topics like project and time management, but also expanded our experiences by pushing us to use the knowledge we have gathered during our time in Bilkent in practical ways.

8. Glossary

Client: Companies and institutions that will be specifying tasks to be completed by Experts using our website.

Expert: Human experts that will be evaluating social media posts based on specified labels using our mobile application.

Label: A metric based on which Experts will be evaluating social media posts. These will be specified by Clients.

Project: Set of specifications, set by a Client, that determines how and when the tasks will be created and distributed.

Task: A unit of work to be completed by an Expert.

Voice Chat: A service that allows Experts to discuss in a group voice call, in real time, regarding a task.

9. References

[1] S. Lohr, “The age of big data,” *The New York Times*, 11-Feb-2012. [Online]. Available: https://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html. [Accessed: 25-Feb-2022].

[2] M. Cavaioni, “Machine learning: Causes of error,” *Medium*, 25-Apr-2017. [Online]. Available: https://medium.com/machine-learning-bites/machine-learning-causes-of-error-87ff372cd5be. [Accessed: 25-Feb-2022].

[3] “Amazon Mechanical Turk,” *Amazon Mechanical Turk*. [Online]. Available: https://www.mturk.com/. [Accessed: 25-Feb-2022].