Bilkent University

Department of Computer Engineering

CS 492: Senior Design Project II

Spring 2022

Low Level Design 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 will be a mobile based platform that aims to provide real time social media analysis by human experts. Clients will be 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 will be available to the clients through our website. The experts will be able to do all of their work through a mobile application allowing them to work remotely and efficiently. Various features will be implemented to ensure the reliability of our experts and provide them the optimal environment to do their work. Gamification techniques will be utilized to incentivize regular work schedules.

1.1 Object design trade-offs

1.1.1 Portability vs. Maintainability

We decided to use the React Native framework to release the mobile application part of our project on multiple platforms. However, sometimes a single codebase may not work properly on all the platforms. This resulted in a requirement to maintain our codebase for each platform separately, thus decreasing the maintainability.

1.1.2 Extendibility vs. Programmability

We tried to use various design-patterns to make the features of our project more expandible. However, this brings about an increased development cost in the short term.

1.1.3 Practicality vs. Functionality

We tried to provide all the essential features and interfaces for our users. However, rather than introducing non-frequently used or trivial capabilities, we gave precedence to the usability of our system.

1.1.4 Robustness vs. Compatibility

Making available a system for various devices increases the number of targeted users. However, the hardware of old devices may fail to perform some necessary tasks. This is why we decided to exclude the devices not satisfying certain requirements to increase the robustness of our applications.

1.2 Interface documentation guidelines

Throughout our project, we have named every construct (class, method, attribute, parameter etc.) using the camel case format. As usual, class identifiers begin with a capitalized letter, while other identifiers do not. Furthermore, we have provided descriptions for our class interfaces using the guideline shown below:

<<class>> ClassName:	
Purpose of the class is explained here.	
Attributes	
-attributeName1 : type of attribute 1 -attributeName2 : type of attribute 2	
Methods	
+methodName1() +methodName2(parameter1 : type of parameter 1)	Purpose of method 1 is explained here. Purpose of method 2 is explained here.

1.3 Engineering standards

UML is a globally used guideline to generate diagrams for software projects [4] that is also taught in our university courses. That is why, throughout the report, we have used UML guidelines

while designing our subsystem composition & class interfaces, and drawing our diagrams. Furthermore, we have used the IEEE system, a universally used referencing style [5], to depict our citations and references.

1.4 Definitions, acronyms, and abbreviations

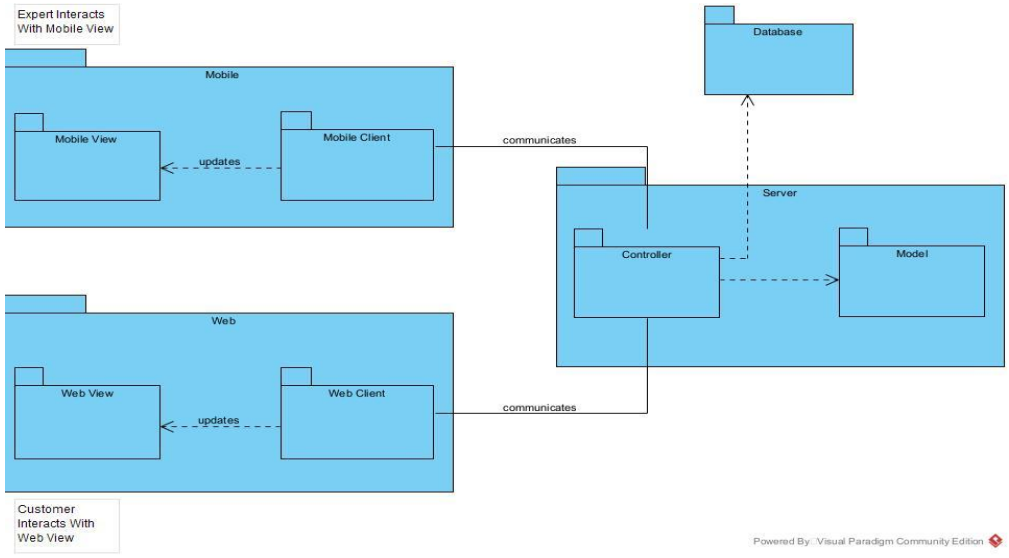
API: Application Programming Interface

IEEE: Institute of Electrical and Electronics Engineers

UI: User Interface

UML: Unified Modeling Language

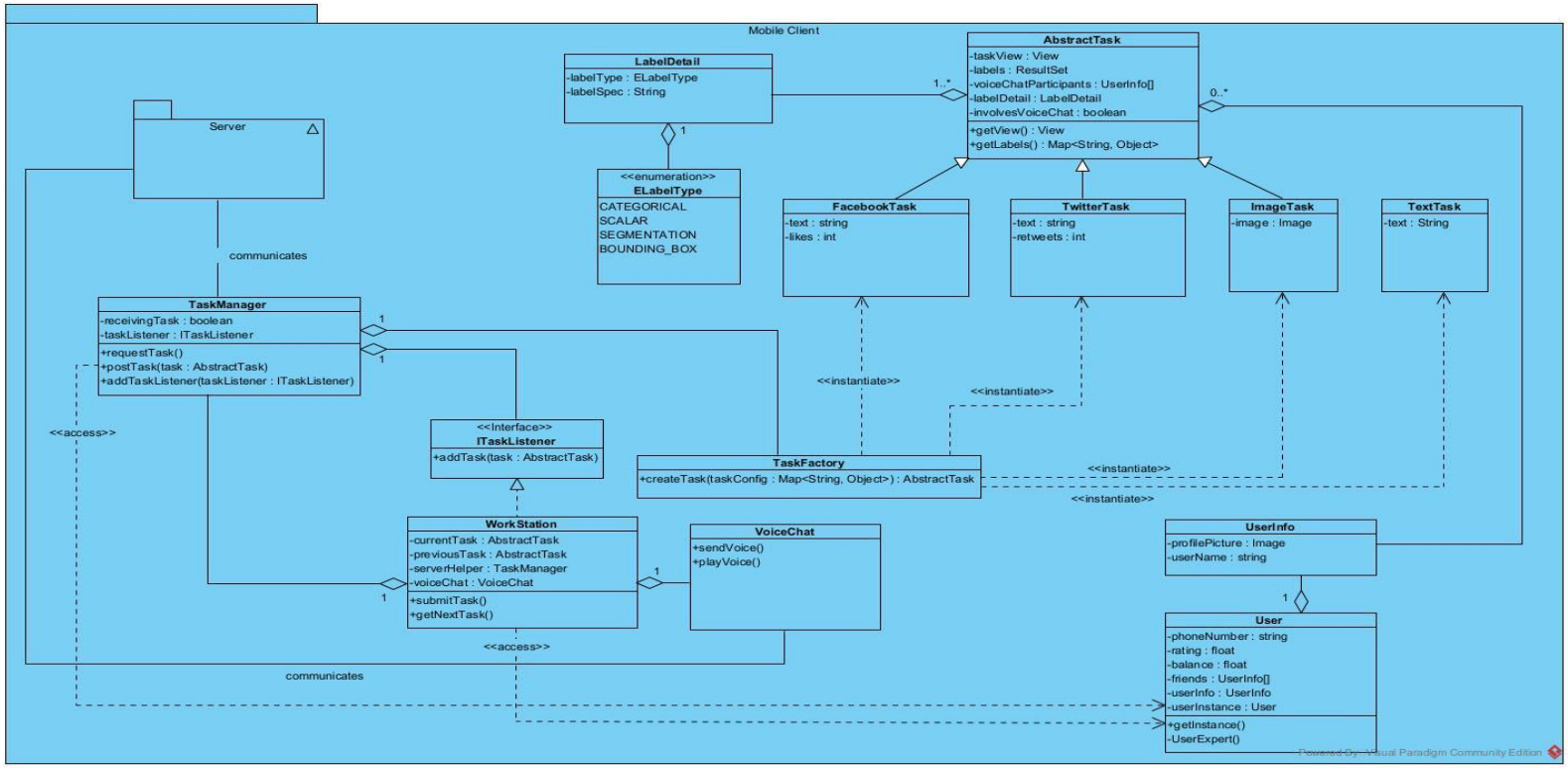
2. Packages



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.

2.1 Clients

2.1.1 MobileClient



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.

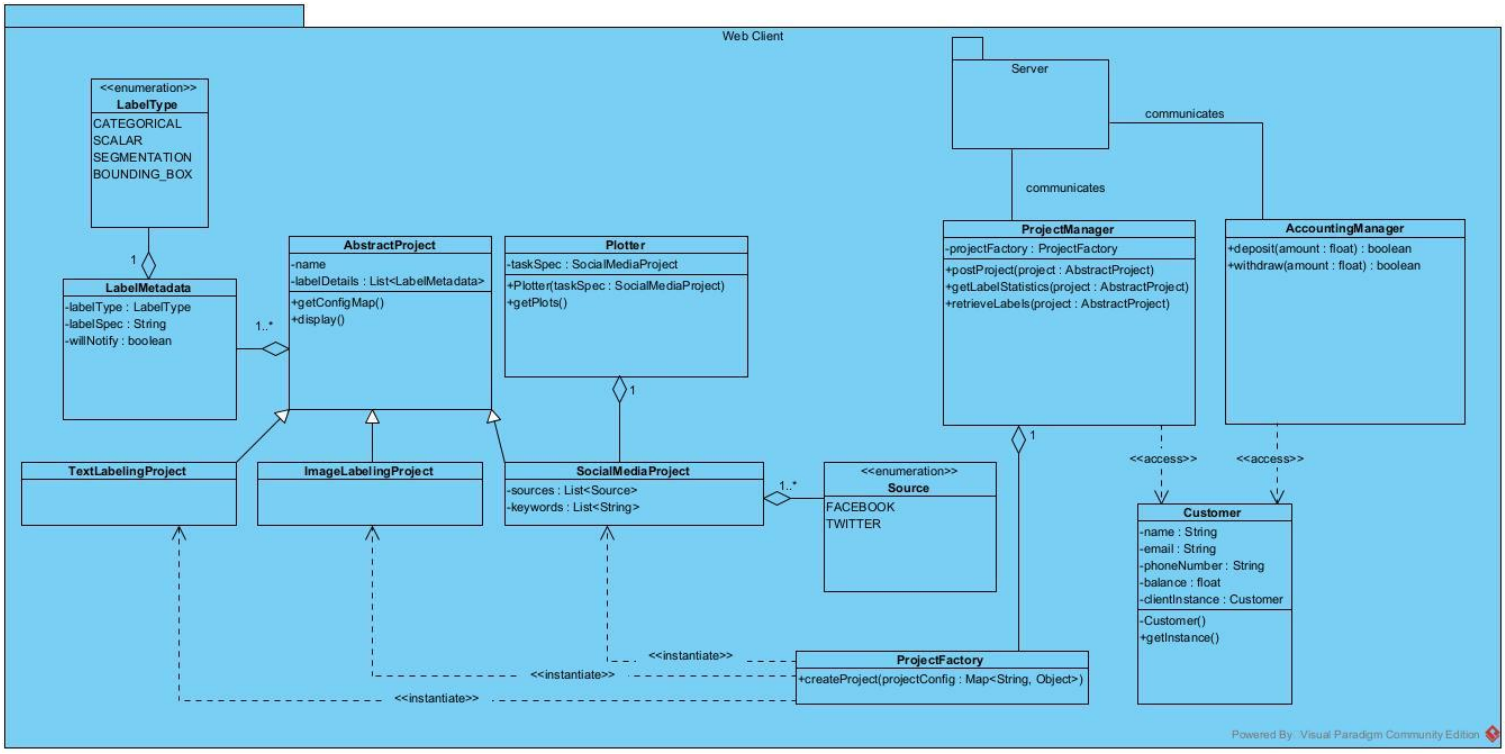
TextTask: It manages plain text data.

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.

2.1.2 WebClient



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.

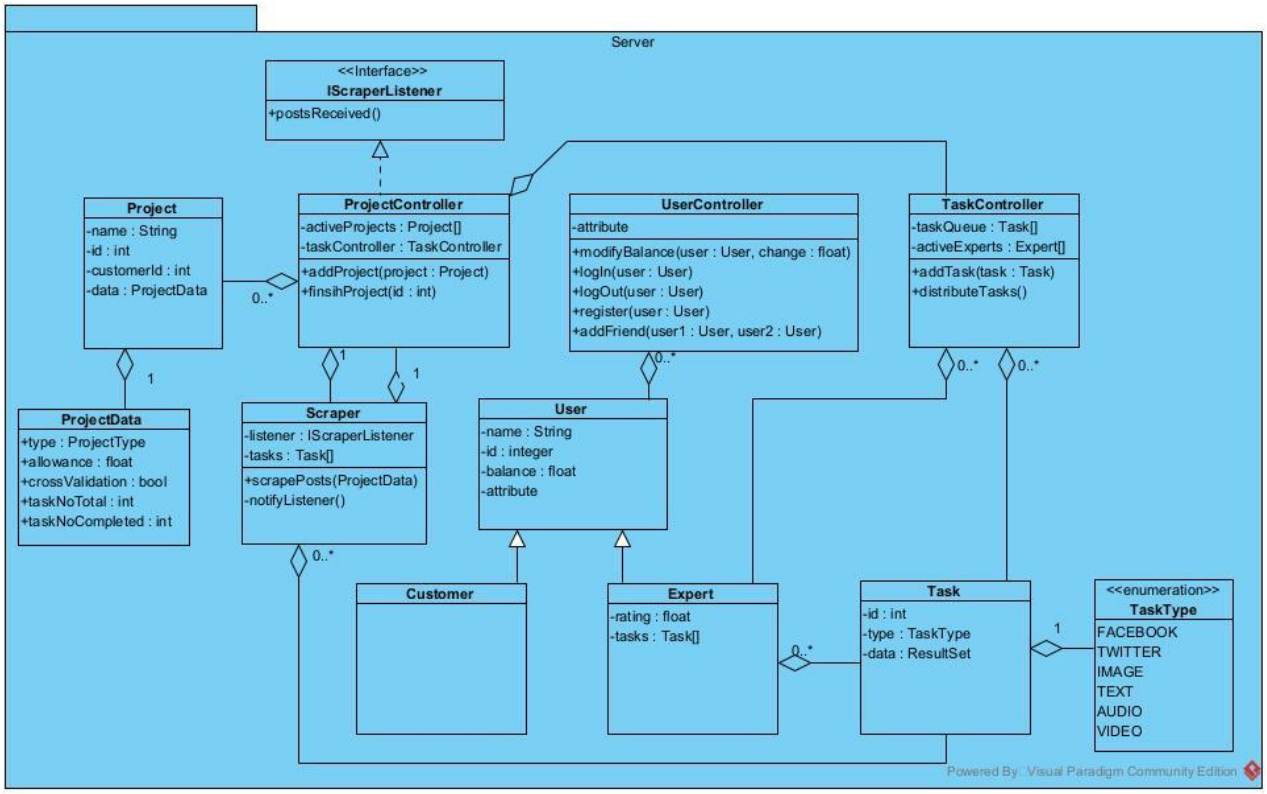
AbstractProject: It manages the configuration related problems while creating a task.

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.

2.2 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. Class Interfaces

3.1 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 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() +getInstance()	Private constructor to apply Singleton Design 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

3.3 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. 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].

[4] “Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series),” *Research Gate*. [Online]. Available: https://www.researchgate.net/publication/234785986_Unified_Modeling_Language_User_Guide_The_2nd_Edition_Addison-Wesley_Object_Technology_Series [Accessed 26-Feb-2022].

[5] “IEEE Reference Guide,” *IEEE Author Center*. [Online]. Available: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf. [Accessed: 26-Feb-2022].