



Bilkent University

Department of Computer Engineering

# Senior Design Project

*Project short-name: CSION*

## Final Report

### **Group Members**

**İsmail Yavuzselim Taşçı**

**Nursena Kurubaş**

**Ayça Begüm Taşçıoğlu**

**Mehmet Sanisoğlu**

**Mehmet Selim Özcan**

Supervisor: Prof. Dr. Uğur GÜDÜKBAY

Jury Members: Prof. Dr. Özgür ULUSOY and Prof. Dr. İbrahim KÖRPEOĞLU

Innovative Expert: Ahmet Eren BAŞAK

# TABLE OF CONTENTS

<b>1. Introduction</b>	<b>4</b>
<b>2. Requirements Details</b>	<b>4</b>
2.1 Overview	4
2.1 Functional Requirements	5
2.2 Non-functional Requirements	7
2.2.1 Usability	7
2.2.2 Efficiency	7
2.2.3 Response Time	7
2.2.4 Reliability	7
2.2.5 Security	8
2.2.6 Maintainability	8
2.2.7 Scalability	8
2.2.8 Recoverability	8
<b>3. Final Architecture and Design Details</b>	<b>9</b>
3.1 Decomposition Diagram	9
3.1.1 Class Diagram	10
3.1.2 NLP Class Diagram	11
3.2 Use Case Diagram	12
3.2.1 Scenarios	13
3.3 Sequence Diagrams	16
3.4 Activity Diagram	18
<b>4. Development/Implementation Details</b>	<b>18</b>
4.1 Frontend	18
4.2 Server/Backend	19
4.3 Decision Algorithm	20
4.3.1 Decision Algorithm for Choosing Category	20
4.3.2 Decision Algorithm for Text Input	22
4.4. Database	23
<b>5. Testing Details</b>	<b>25</b>
5.1 Algorithm Testing	25
5.2 Frontend/Client Testing	26
5.3 Backend/Server and Database Testing	27
<b>5.4 NLP Testing</b>	<b>28</b>
<b>6. Maintenance Plan and Details</b>	<b>28</b>
<b>7. Other Project Elements</b>	<b>30</b>
7.1.Consideration of Various Factors	30

7.2.Ethics and Professional Responsibilities	31
7.3.Judgements and Impacts to Various Contexts	32
7.4 Teamwork and Peer Contribution	34
7.5 Project Plan Observed and Objectives Met	35
7.6 New Knowledge Acquired and Learning Strategies Used	35
<b>8. Conclusion and Future Work</b>	<b>36</b>
<b>9. Appendix - User Manual</b>	<b>38</b>
9.1 Csion Application Usage	38
9.2 Csion Mobile Application Installation	46
<b>References</b>	<b>49</b>

# 1. Introduction

Throughout life, people are bound to experience certain ups and downs, as such is life. In several cases, we come across a type of crossroads, in which we try to find the optimum outcome. However, it is not always easy to take every aspect of the problem into account and make a rational decision. In such cases, the advice of a friend is always welcome as it provides a different view. However, the advice you get is not exactly bias-free let alone optimal for the situation you are in. *Csion* however, relies on your inner self, as well as your current mindset and as such, is a reflection of yourself, giving you advice in an objective, rational and personalized manner.

The following parts provide detailed information about the overall final structure of our project with model specifications and requirements via our models. The models provided define the system we propose with class, activity and sequence diagrams. Next, we have provided insight on our development stages and choice of implementations. We go into detail in our testing strategies and the cases we have covered after which we discuss our proposed strategy to maintain this project. We discuss our future expectations and improvement ideas. In the last part, we have included glossary and references sections to complete our report.

## 2. Requirements Details

In this section we will explain our functional and non-functional requirements.

### 2.1 Overview

*Csion* aims to optimize people's decisions about any subject based on their characteristic behaviours within a short amount of time. We are using Myers-Briggs Personality types to have an initial categorization of the users' characters [1]. This test is highly standardized and is esteemed in the field of psychology. There are 16 unique personalities and each person can find their personality type by solving simple and quick test questions. These types have certain borderline character traits that generally conform to the way people behave. We use the "Crystal Know" API to implement this test and get the results to use inside our application.

Although it is a deciding factor, personality alone is not the only variable when a person makes a decision. A detailed analysis of the subject that the user will decide on, is also needed to provide reliable suggestions. For this end, in our project *Csion*, we have implemented a Chatbot which uses Natural Language Processing techniques to give detailed analysis of a sentence or paragraph. We expect from users to write the problem itself as a text. Then by combining that analysis and the

person's general personality, we give suggestions to our users to help them decide. However, this method is not a robust way of providing suggestions as the performance of our implementation depends highly on the inputted text. Nevertheless, we do provide a broad analysis via this technique.

Another way of using Csion is through our question-answer section. In that section, users are only able to get suggestions from predetermined categories such as Relationship, Career and Education. From these categories they navigate to their subcategory like "Romantic", "Change Career" or "Study Field". Users are prompted to select a specific problem that best describes their situation in their selected subcategories like "Should I change my field?". After selecting the problem, we ask predetermined questions to the user to get important information that affects the decision for that problem. After getting answers from users, we combine the personality of the user and the answers that we get to determine what users should decide. After this processing phase, we again show a detailed suggestion to the user. To increase the usability of this section, questions include only yes or no to questions.

After giving suggestions, we ask the user to give feedback on whether or not he chose what we suggested, and his/her satisfaction rate about the decision. With using this feedback data and deep learning implementation, we enable our application to improve itself and be more personalized to make better suggestions for future questions.

## 2.1 Functional Requirements

This part is about the functions of the application that affects user experience.

### **Sign-Up/Log-In Process**

When users open the application for the first time after downloading, they will encounter a page with two options: Login and Sign Up. Users have to register to use the application, but if they are already registered, they can just click the "Login" button and login by typing the required information and clicking "Login".

To register to the application, users need to click "Sign Up" and fill the boxes that require basic personal information such as name, surname, e-mail etc.. After completing the registration form, as the last step of registration, users should solve a personality test. Afterwards, the system will automatically login the user to the application.

### **Asking for a Decision**

When a new session is created, the user will see the Main Page and it will have two different tabs: "Choose Category" and "Type Problem". It is recommended for users to choose the Category system

if their questions have a match there, but for all other complex questions, they are free to use the Chatbot system.

If users choose the first tab, they will be able to choose between different categories. When they click a category, subcategories of that category will open and users will select the category that is related with their doubts. After selecting the desired problem, the system will start to ask users several questions related to the topic to increase the accuracy of the final decision. Users are able to respond to the questions by selecting “Yes” or “No”. After completing all the questions, the final suggestion that is prepared specifically for the user will be presented at the analysis page.

If users choose the Chatbot tab, they will see a form. Users should type their question, and pick a deadline and it is recommended for them to provide as much detail as possible about their problem, in order to get more accurate results. After submitting the form, users will see the analysis page.

## **Profile**

Each user has a profile that displays their personal information, personality types that are obtained from tests, a “Former Q/A” section that users will be able to see their former decisions, the advice they took from the application, the decision they made and how content they were with this decision. They are also able to give feedback to their former questions through this page.

## **Giving Feedback**

When the user enters his/her profile, they will be able to give feedback on what they chose to do and how much they are content with their decision to increase the accuracy of their next problems.

## **Settings**

Users are able to reach the settings page through a button at the navbar that remains at the main page. Settings page includes account settings, like changing password, suspending the account or notification settings, an “About Us” section for users to have knowledge about the contributors of the application, a “Log Out” section to enable them to end their session.

## 2.2 Non-functional Requirements

In this section we have listed primary non-functional requirements for Csion.

### 2.2.1 Usability

- Application should be understandable for every user. User interface provides every information that user can access.
- Menus and sub-menus are usable for every portion of the application. Also, menus are easy to tap and easy to read since users can have difficulties for reading.

### 2.2.2 Efficiency

- Number of questions does not exceed 10 for categorized problems.
- Questions cover at least 1 keyword of the personality type in order to decrease the number of questions.
- Analysis output should be direct and simple.

### 2.2.3 Response Time

- Answers of the questions and any inputs of the user are processed in less than 1 second.
- Data transfer between Database and server lasts less than 1 second for login and signup operations.
- Communication between servers lasts less than 1 second.
- Application creates a final analyzed output less than 1 second.

### 2.2.4 Reliability

- Application provides exact results of personality tests without any errors.
- Result of analyzed text that comes from IBM Watson returns acceptable keywords.
- Application stores each answer of questions correctly and their value as a variable for the analysis algorithm. Algorithms work coherently for the same inputs.

### 2.2.5 Security

- The answers of questions are accessible only by the user should they decide not to give feedback.
- Outside of the application, the results of personality tests are only accessible by Crystal Knows since Crystal Knows has the right to keep the results of tests.
- Any given personal information such as passwords and social platform information are encrypted and protected that only user and application developers can access.
- Personal data of the users are deleted if the users delete their account.

### 2.2.6 Maintainability

- Application is easy to fix for errors less than an hour in case of any wrong output
- Application does not encounter any errors in the integration of different APIs and platforms.

### 2.2.7 Scalability

- Database is scalable upto 1000 users for the deployment stage of our project.

### 2.2.8 Recoverability

- Database manipulations take place in sequence and can be repeated in case of a failure.



## 3. Final Architecture and Design Details

### 3.1 Decomposition Diagram

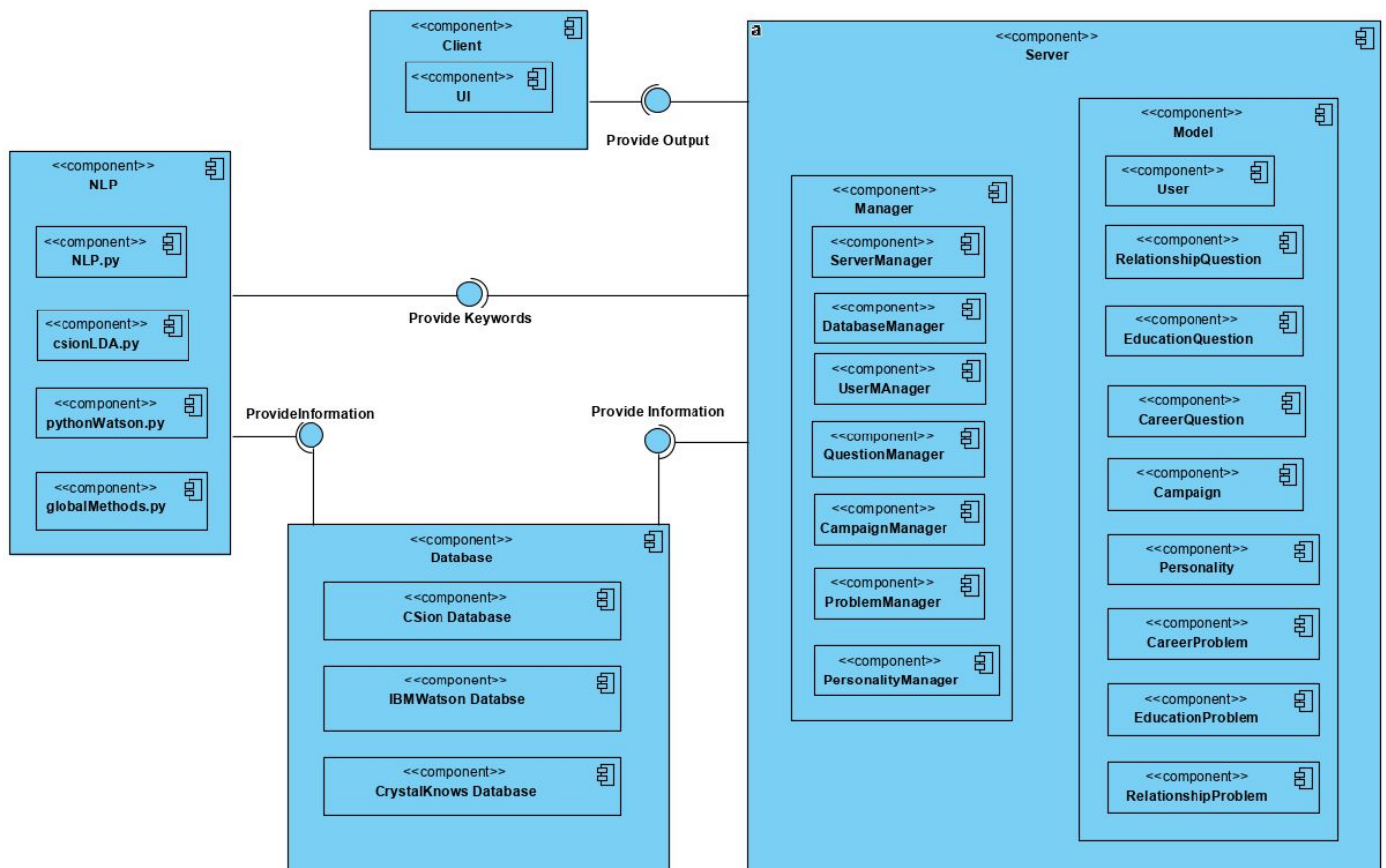


Figure 1: Decomposition Diagram

### 3.1.1 Class Diagram

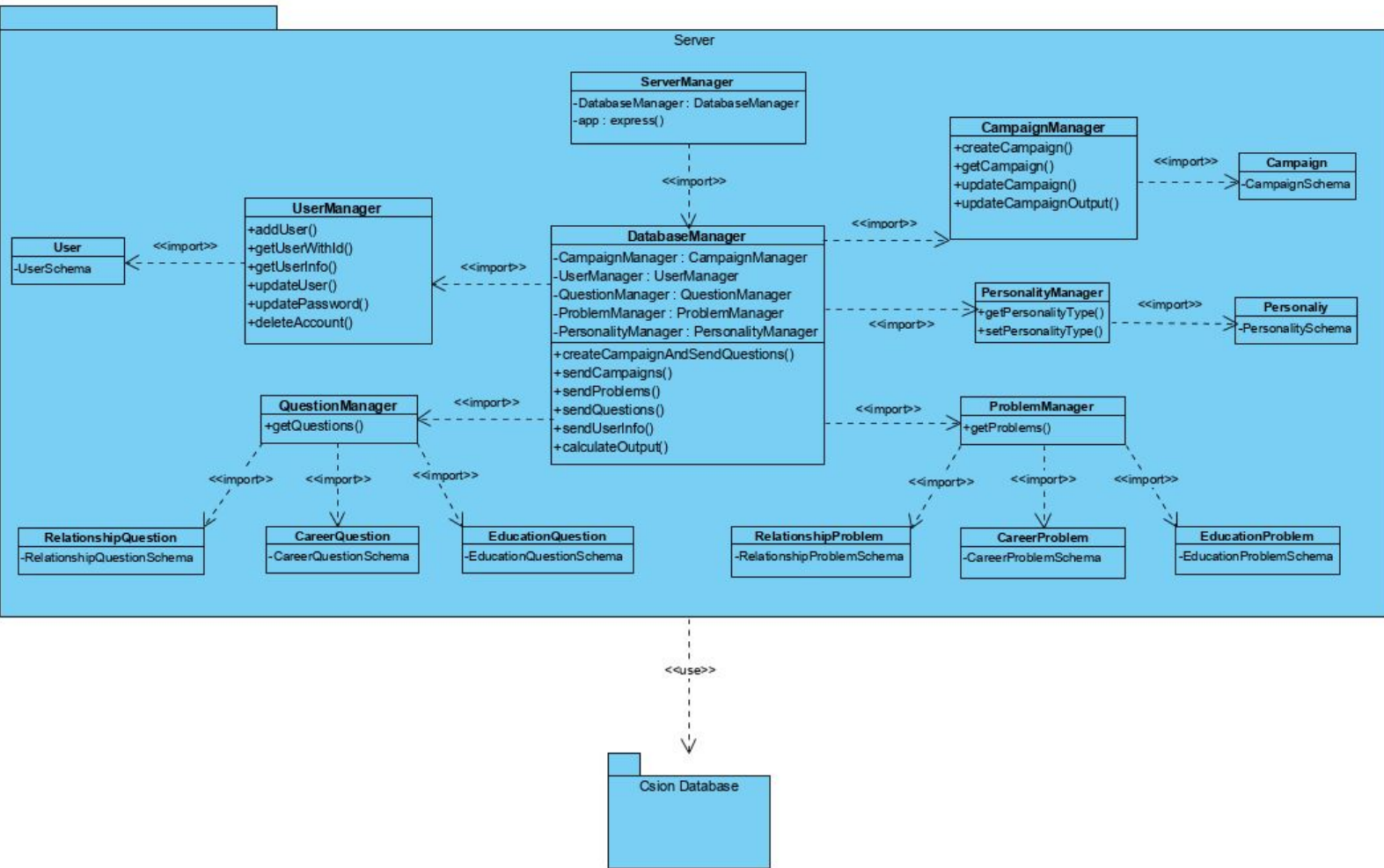


Figure 2 : Server Class Diagram

In the server side we have constructed a structure such that each manager will be responsible for some part of the data flow. Schemas are the instances of the tables according to Csion Database. Since we have used mongoose, we need these instances for any manipulation that we want to execute [2]. Summary of these classes is following:

- **ServerManager** will handle requests from Client side and send the responses back. It has an “app” attribute which is an instance of express framework and it listens to every request from Client side by creating several endpoints.
- **DatabaseManager** is responsible for invoking the necessary functions of other Managers according to requests. It is the first and the last stop point for the backend side.
- **UserManager** is responsible for Csion Database interactions that involve account manipulation by using UserSchema that is created in **User** class.

- **QuestionManager** is responsible for Csion Database interactions that involve any kind of question gathering for the Client. It uses 3 schemas from **RelationshipQuestion**, **CareerQuestion** and **EducationQuestion** classes.
- **ProblemManager** is responsible for Csion Database interaction that involves any kind of problem gathering for the Client. It uses 3 schemas from **RelationshipProblem**, **CareerProblem** and **EducationProblem** classes
- **CampaignManager** is responsible for Csion Database interactions that involve campaign manipulation whenever Client asks for questions or answers them. It uses CampaignSchema from **Campaign** class.
- **PersonalityManager** is responsible for Csion Database interaction when a user finishes the personality test or s/he goes to the profile page. It uses PersonalitySchema from **Personality** class.

### 3.1.2 NLP Class Diagram

In this section, you will find our NLP class diagram and its explanation.

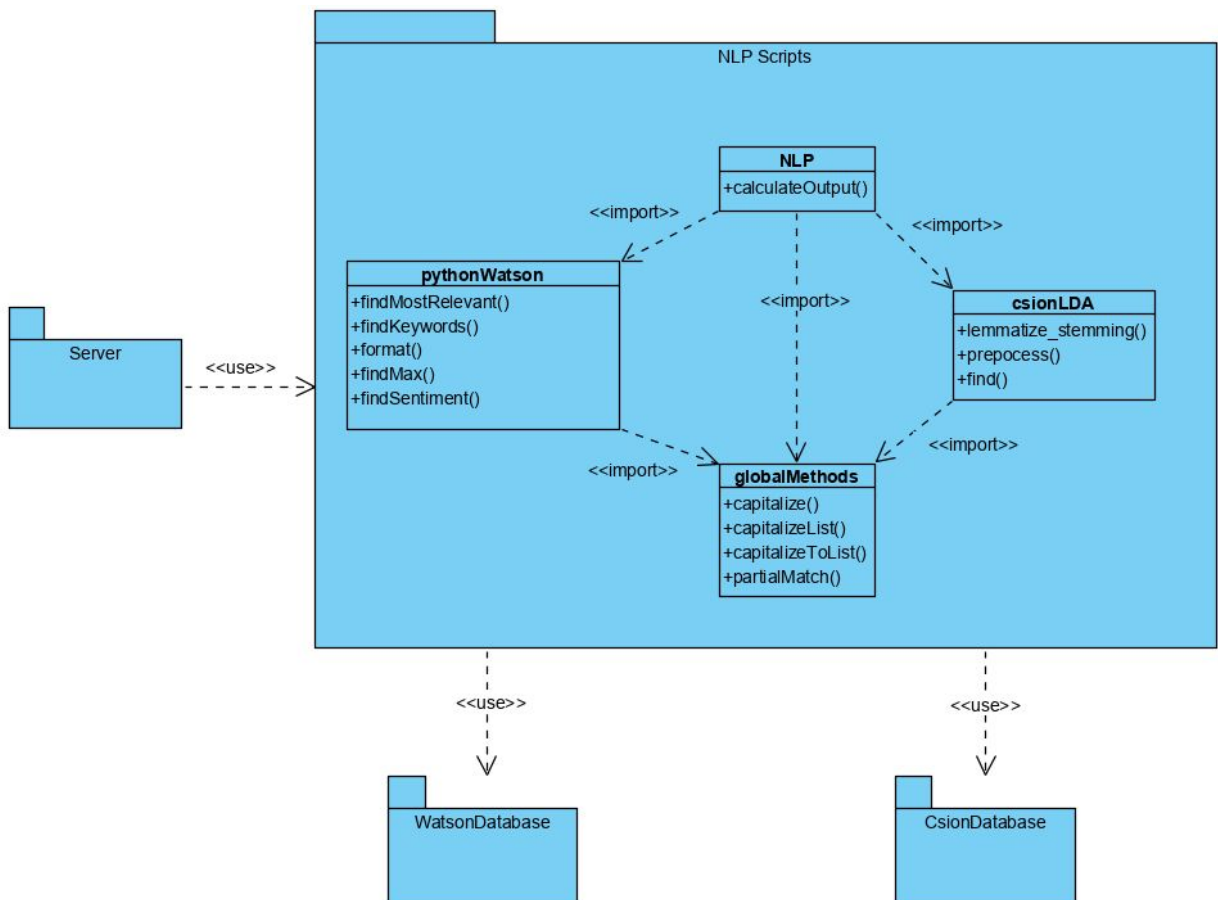


Figure 3 : NLP Class Diagram

As a part of our implementation, we have used several python scripts for interacting with IBM Watson in order to implement the NLP functionalities. When a user wants to express their problem via writing instead of choosing from defined categories, our server invokes the pythonScripts and receives keywords after several functions.

- **NLP.py** imports other scripts and calls their functions. It also finds personality attributes of the user from the Csion Database and evaluates the final score according to keywords match. It will be detailed in section 4.3.2
- **pythonWatson.py** receives the text and sends a curl request to Watson API for retrieving keywords.
- **csionLDA.py** receives the text and implements stemming and lemmatization algorithms on it. Then it loads our ML model and predicts its category. According to category, it retrieves the most relevant keywords of that category.
- **globalMethods.py** is a helper file that consists of global functions which we are using for other scripts.

### 3.2 Use Case Diagram

At this part, you can find our updated use-case diagram and user scenarios in detail.

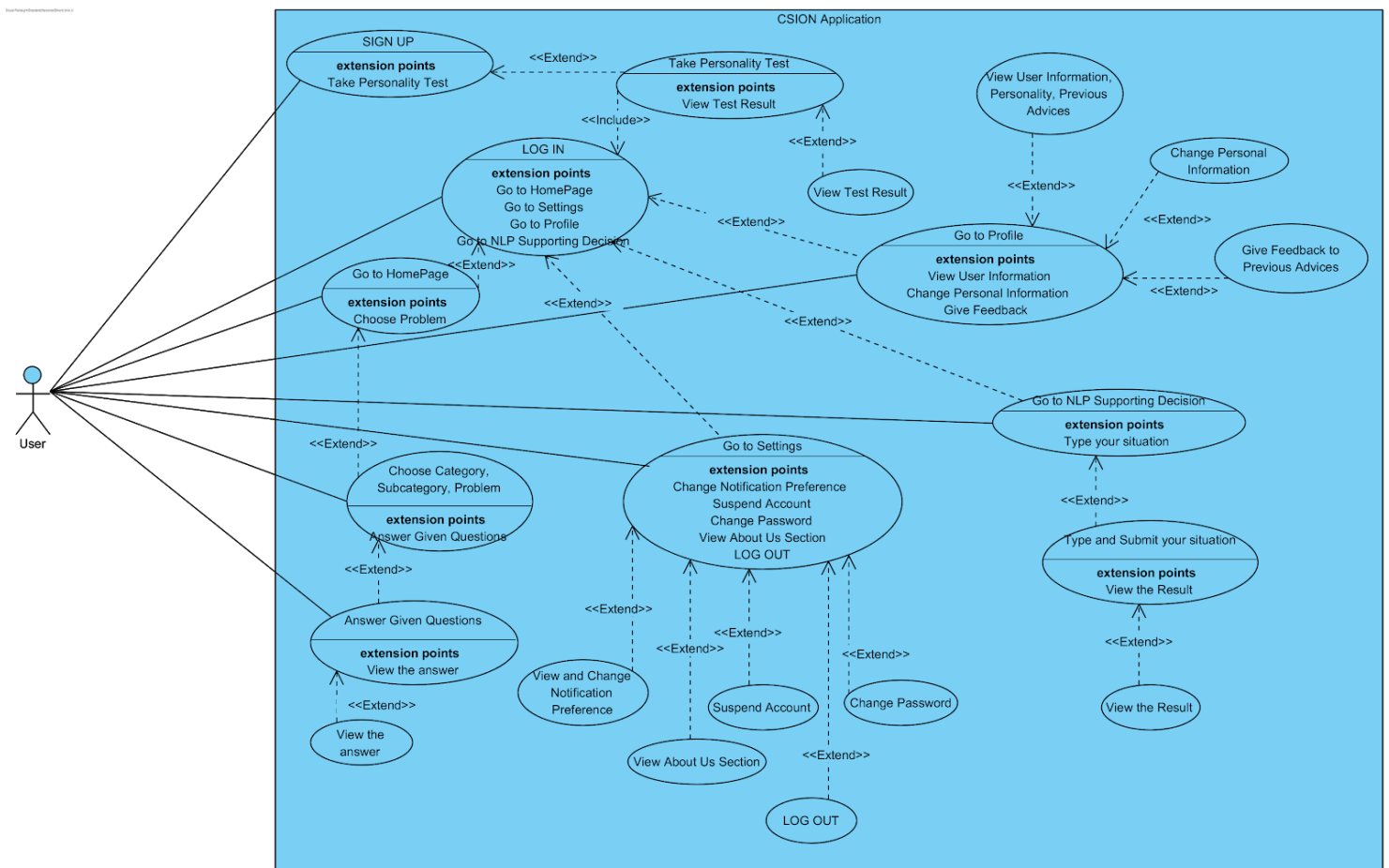


Figure 4. Use Case Diagram

### 3.2.1 Scenarios

In this section you can find our user scenarios.

#### **Sign Up**

Actors: New user

Conditions: User opens the app and click sign up button or from login page clicks the sign up link

Exit Conditions: User closes the app or clicks log in link.

Flow of Events:

1. User clicks "Sign Up" .
2. The registration page is open.
3. User specifies the account's username, email and password.
4. System creates a new account for the user and logs in the user to the application.

#### **Log In**

Actors: Registered User

Entry Conditions: User clicks "Log In".

Exit Conditions: User clicks "Log Out".

Flow of Events:

1. User clicks "Log In" on entrance page.
2. User types username and password.
3. User clicks "Log in" button.
4. System checks the given username and password pair with the pair in the database.
5. If matches, the home page is opened; if not, the user proceeds on Step 2.

#### **Take Personality Test**

Actors: New Registered User

Entry Conditions: New registered user log in.

Exit Conditions: User closes the application or clicks continue.

Flow of Events:

1. User is redirected to the Personality Test Screen.
2. User fills a personality test and submits.
3. System detects and stores personality type of user.
4. Personality details are shown to the user.

#### **Using Categories to Get a Decision**

Actors: Logged in user

Entry Conditions: User logged in and in the Home page.

Exit Conditions: User navigates to another page.

Flow of Events:

1. User selects a category.
2. User selects a subcategory or click back button to go to Step 1.
3. User chooses a problem or clicks the back button to go to Step 2.
3. User sees a form, fills and submits it.
4. System runs the algorithm and the user gets a decision to use with some details.

### **Using NLP Supporting System to Get a Decision**

Actors: Logged in user

Entry Conditions: User clicks NLP tab.

Exit Conditions: User navigates to another page.

Flow of Events:

1. User clicks NLP tab.
2. User types an explanation about the situation and his/her question in a text field in at least 40 characters and clicks Submit.
3. NLP Algorithm runs and shows a decision to the user with some details.
4. Users can go back to Step 2 by clicking the Back button.

### **Change Personal Information**

Actors: Logged in user

Entry Conditions: User clicks Profile tab.

Exit Conditions: User clicks a tab other than Profile.

Flow of Events:

1. User clicks Profile tab.
2. User views the personal information.
3. User clicks the edit button.
4. User changes the information and clicks Submit.

### **Give Feedback**

Actors: Logged in user who got a decision before and did not give feedback

Entry Conditions: User clicks Profile tab.

Exit Conditions: User clicks a tab other than Profile.

Flow of Events:

1. User clicks Profile tab.
2. User views the give feedback button under previously asked questions who did not get any feedback yet.
3. User clicks a give feedback button.
4. User fills the form.
5. User clicks Submit and the system saves the feedback.

## **Log Out**

Actors: Logged in user

Entry Conditions: User clicks Settings tab.

Exit Conditions: User log in to the system.

Flow of Events:

1. User clicks Settings tab.
2. User scrolls down and clicks Log Out.
3. User is navigated to the entrance page.

## **Change Password**

Actors: Logged in user

Entry Conditions: User clicks Settings tab.

Exit Conditions: User navigates to another tab.

Flow of Events:

1. User clicks Settings tab.
2. User views the Change Password form.
3. User types current password and desired new password.
4. User clicks Submit and the form becomes empty and the system changes the password.

## **Delete Account**

Actors: Logged in user

Entry Conditions: User clicks Settings tab.

Flow of Events:

1. User clicks Settings tab.
2. User views the Delete Account form.
3. User types current password.
4. User clicks Submit and the system logs out the user and deletes the account for good.

### 3.3 Sequence Diagrams

The followings are the sequence diagrams for the basic functionalities of Csion.

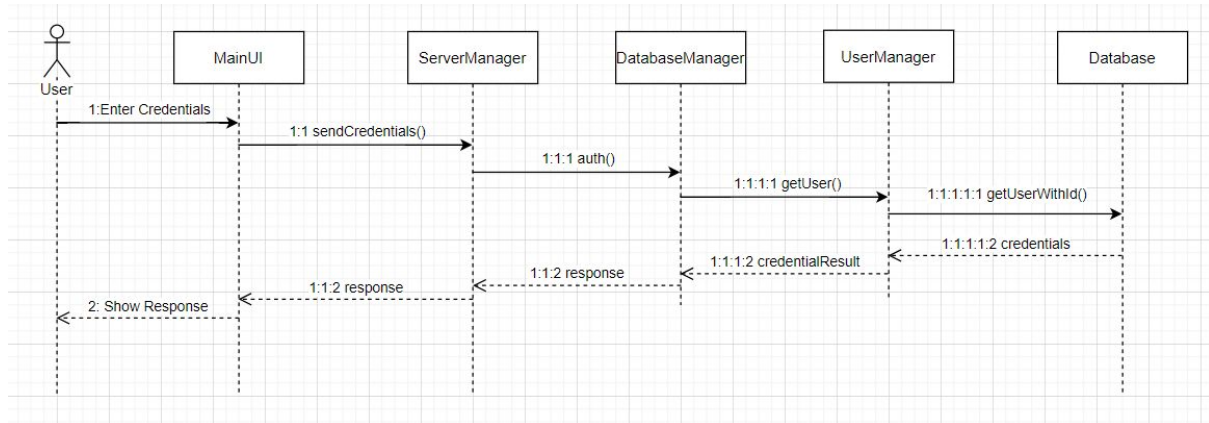


Figure 5. Sequence Diagram for Problem Login Action

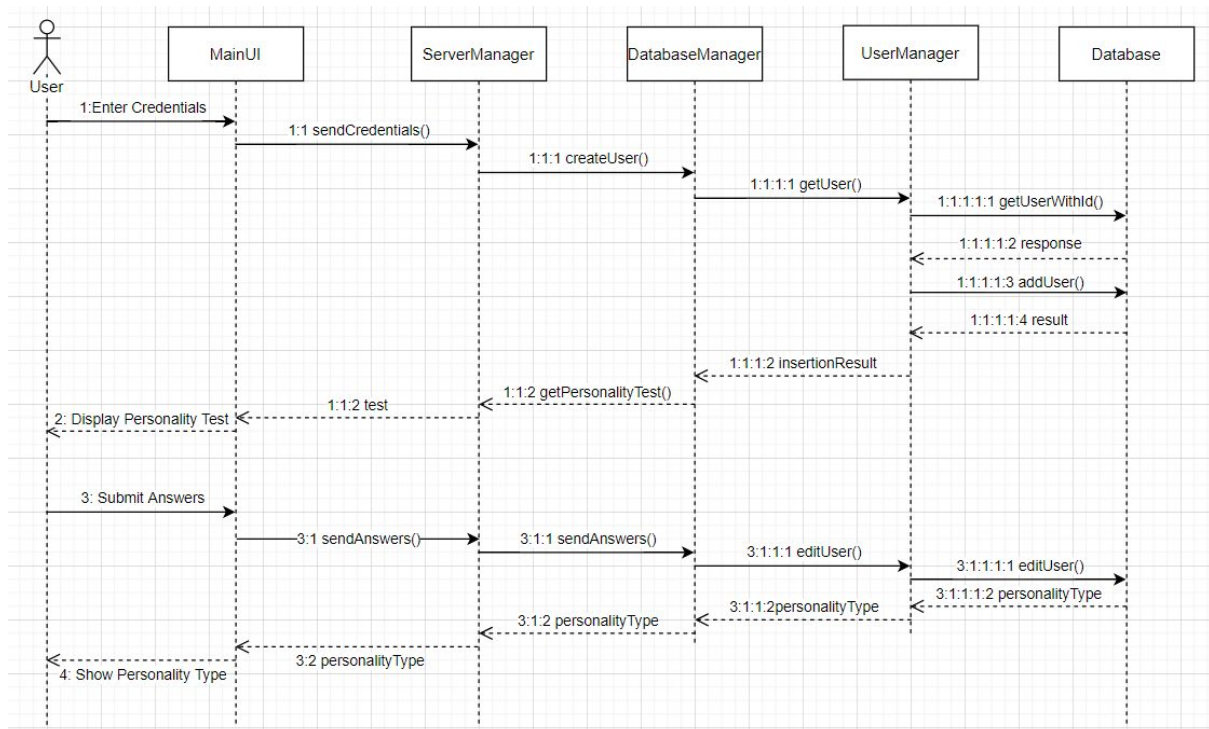


Figure 6. Sequence Diagram for Register and Submit Test Actions



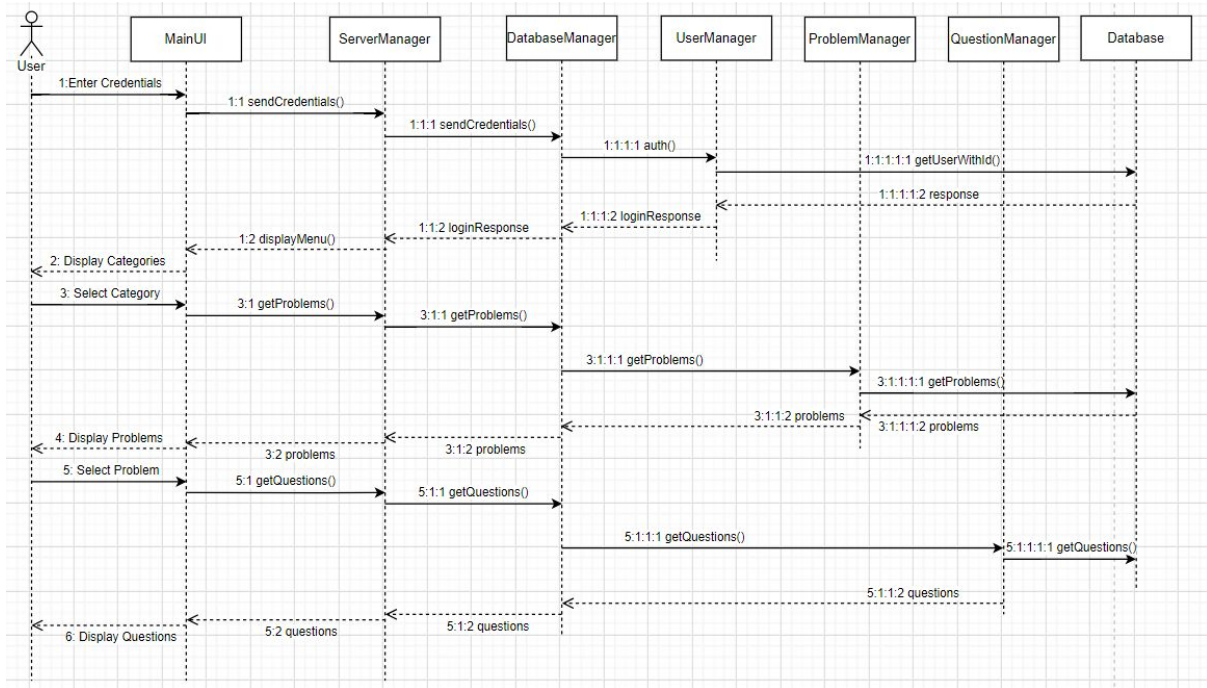


Figure 7. Sequence Diagram for Selecting Category and Answering Questions Actions

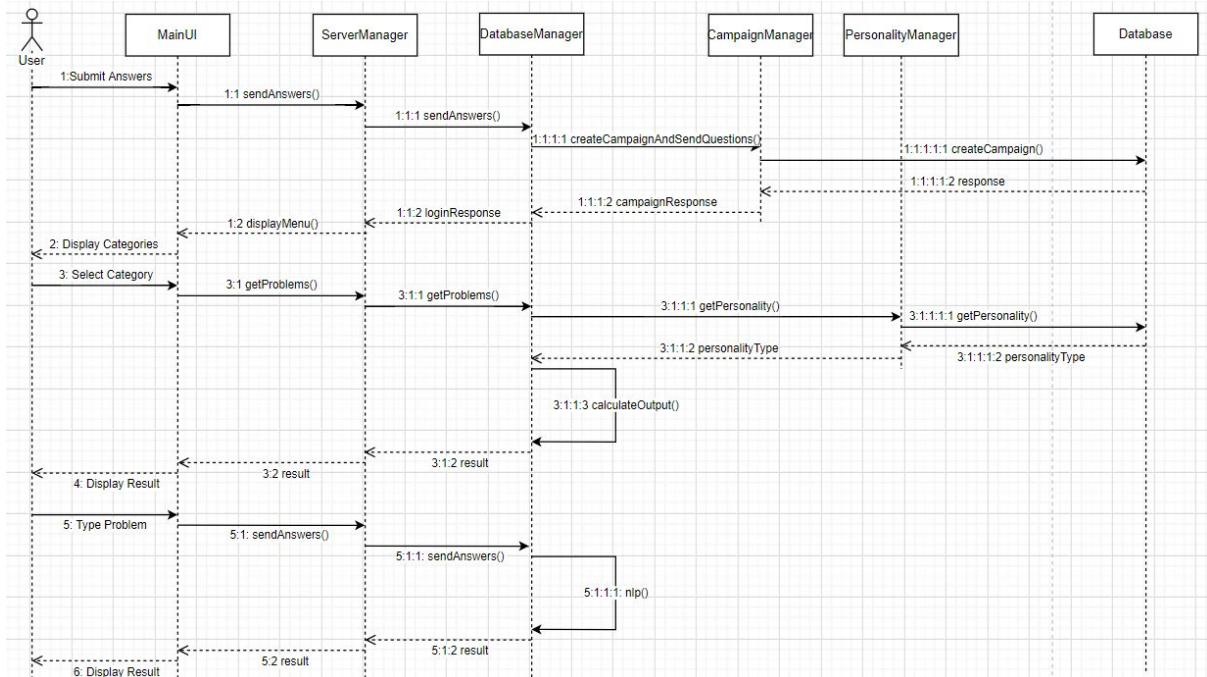


Figure 8. Sequence Diagram for Problem Selecting and Problem Typing Actions

## 3.4 Activity Diagram

The final diagram is the Activity Diagram. We have explained the most general data flow when user interacts with our application.

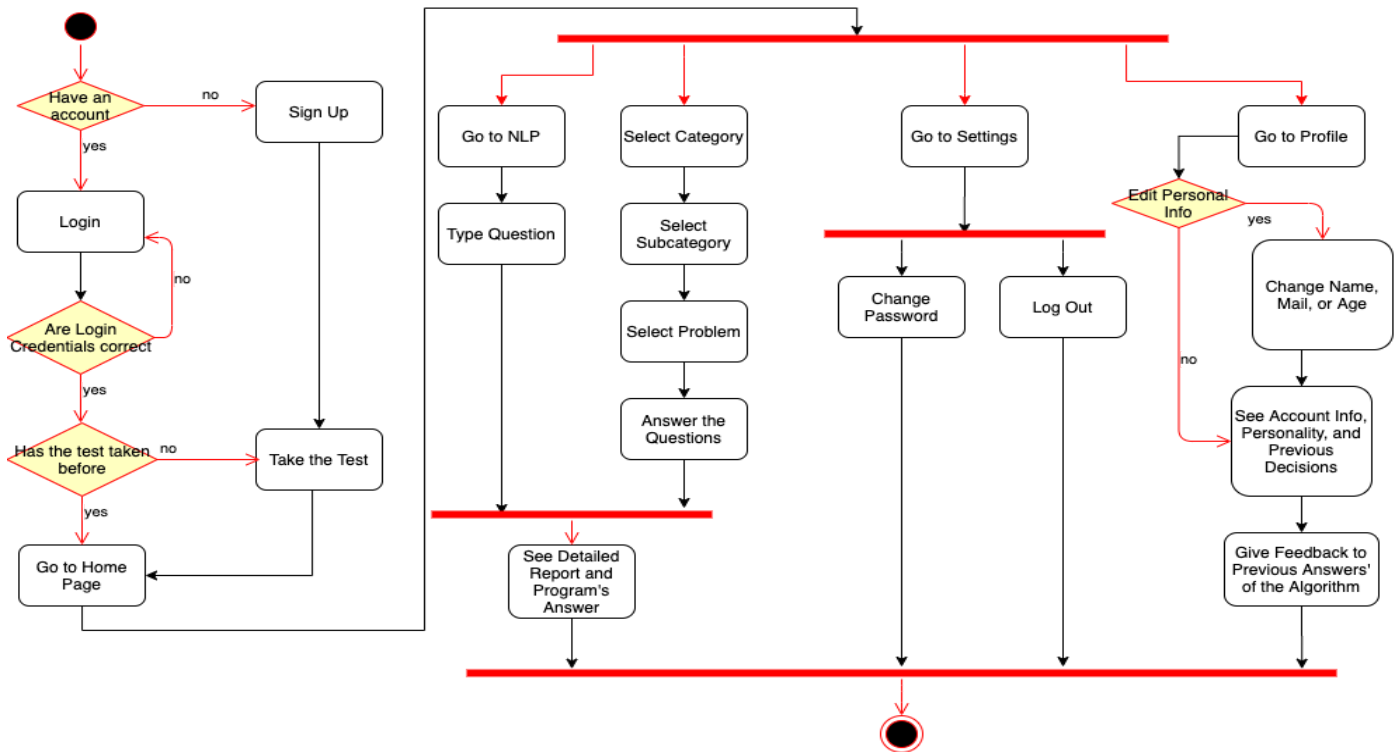


Figure 9: Activity Diagram of Csion

## 4. Development/Implementation Details

In this section, you can find development and implementation details of our application. We talked about frontend, backend, and our decision algorithm in the order of mentioning.

### 4.1 Frontend

Frontend is the part of our application where all the client part resides. Here, we implemented the user interface of the application, managed the user interactions with the application by also communicating with the server through a REST API. Although we are developing a mobile application, we decided to implement it in a web-based manner and thus, we developed the frontend of the application by using HTML, CSS, JavaScript and JQuery technologies. The reason for this preference was that we believed that it would be faster this way. In fact, at the end of the development of the first version, we can say that it was the right choice because we were very familiar with the web technologies and it helped us to both develop and fix the bugs faster than we would in a mobile developing environment.

We developed the frontend by using Visual Studio Code, since it is also compatible with Git, and after the coding finished, we moved the code to Glitch environment since testing with server interactions would be easier there. Still for debugging and further developing, we use the Glitch environment [3].

Our first frontend implementation experience started with using Angular Framework, which we believed that it would help us build a component-based object-oriented project. However, it blocked the mobile-testing environment that we used at the beginning, which was ngrok, so we decided not to use a framework while developing frontend [4].

At first, we developed each page in a different module and tried to come up with a component-based implementation where each page has its own route, HTML, CSS and JS file. However, at the testing phase, we realized that although it is working pretty well in the desktop environment, it was causing several issues at mobile version. The most important drawback was that we did not want users to see the browser components for a full mobile-application experience. However, although we hide the browser components for the main page, each time the route changes, browser components become visible again. And thus, we converted the program to a one-component program by gathering all page elements such as HTML and CSS files into one file each (index.html and index.css), and we arranged the transitions between pages in such a manner that navigation was happening by hiding and showing html elements. This arrangement was also advantageous for our navigation feature, for us to keep the pages' current situation when the user changes the page through the navigation bar.

For the implementation of the personality test, we get use of Crystal-Knows API for developers since this test gives a reliable result in a form that is suitable for the application [5]. We show the results to the user right after the test, and at the profile page.

## 4.2 Server/Backend

In our server implementation, we choose to use modern languages and frameworks. We implemented all our servers with Node.js except the NLP module which is implemented by using Python. For creating robust, scalable and fast servers, we choose to use Express.js which is a framework over Node.js. What it does basically, simplifies the server and endpoint creation. We chose Express.js because it is a battle-tested framework and it is very easy to use. About the database choice we decided to use a NoSQL database which is MongoDB. We choose MongoDB because it is very well integrated with Node.js already and it is a very flexible database which we might need. Also we already have some knowledge about it, so overall it was a great choice.

After explaining our technologies, let's look at the structure of our server. At the heart of our server, we have a REST API which can be interacted with frontend easily. While developing the frontend, when we see that we need to get information from the server, we created the corresponding endpoint in the server. Because of this approach our frontend and backend grew at the same time and we did

not implement unnecessary features in the server or in the frontend. After creating endpoints we populate those endpoints in a way that frontend requires. For example, when using the NLP feature our frontend makes a request to an endpoint with a problem that the user typed, and our server runs our Python program with the user input and gets the output. Then it organizes that output in a way that frontend wants and lastly it sends the resulting object.

We developed the backend by using Visual Studio Code, since it is also compatible with Git, and after the coding finished, we moved the code to the Glitch environment since testing with client interactions would be easier there. Still for debugging and further developing, we use the Glitch environment. Also right now, we bought Glitch Boost to increase our disk space, ram, and cpu of our server. Also this boost enables our server to become awake all the time. Because of this reason, Glitch satisfies our hosting requirement so we decided to stick with the Glitch.

## 4.3 Decision Algorithm

The decision algorithm is the backbone of our entire project. We need to have a consistent, accurate and improvable algorithm. In order to achieve such an algorithm, we have done a great amount of testing as detailed in Chapter 6. For creating a detailed output, we have followed two different approaches for both Choosing Category and NLP options.

### 4.3.1 Decision Algorithm for Choosing Category

We will explain how this algorithm works step by step:

- User select category, sub-category and problem
- User answers the several questions and indicates the desire level
- Client side sends the following parameters to server:
  - An array of “1” and “-1” that holds the answers of the question. 1 means yes and -1 means no. Any “yes” answer encourages the user to do something about the problem and vice versa. For the sake of simplicity this array will be referred to as “results”.
  - CampaignId for fetching the corresponding campaign that has already been created during the selecting problem process. Campaign holds information about current process such as category, sub-category, problem, questions, answers, ratio.
  - DesireLevel that is between -10 to 10.
- Algorithm gets these parameters along with accountId from the session.
- User gets found by using getUserWithId function with parameter userId
- User’s personality type’s attributes get found by using getPersonalityType function with parameter personalityType string.
- Campaign gets found using getCampaign function with parameter campaignId
- Problems, questions and their attributes get copied by using Campaign.

- Each question's score gets multiplied by the element of the "results" array and they get added up.
  - For instance if the results array is [1,1,-1,1,-1] and scores of questions are [6,5,8,9,2] the answers score will be  $(6*1)+(5*1)+(8*-1)+(9*1)+(2*-1) = 10$
- Desire level gets added up
- The ratio for the answers of the questions gets calculated according to this formula
  - $\text{answers\_ratio} = (\text{answers\_score}) / (\text{full\_score}) * 50$
  - full\_score is the maximum value of answers\_score can be. For the above example:
    - Let desireLevel = 10. Then final\_score = 20, full\_score = 40 and answers ratio = 0.5
  - The reason for multiplication with 50 is, scores from answers will affect only half of the output. The other half is the ratio for the personality keyword matching score.
- The matching encourages keywords to be found. Matching encourage keywords means a user has such a personality attribute that encourages him/her to do something about the problem.
- The matching discourage keywords get found. Matching discourage keywords mean a user has such a personality attribute that discourages him/her for any kind of action.
- The ratio for the personality keyword matching score gets calculated according to this formula
  - $\text{keyword\_ratio} = (\text{COEFFICIENT}) * (\text{matched\_encourage} - \text{matched\_discourage}) * 50 / (\text{total\_keywords\_length\_of\_personality\_type})$
  - COEFFICIENT is some number that can change according to feedback of our users. Currently we have assigned it to 5 since we have observed this number gives enough results for the early stages of the application.
  - The reason for multiplication with 50 is the same reason that we have explained above steps. This ratio is the other half of the output
- Campaign gets updated according to total ratio
- Server sends following parameters to Client
  - An array that holds matched encourage keywords
  - An array that holds matched discourage keywords
  - Total ratio which is between 0 to 100
  - Personality type of the user.
- User sees the output
  - The ratio of the output indicates the decision of our algorithm Since our problems consist of "Should I.. ?" type of questions, we are showing the decision according to:
    - 0-25 >> User should definitely not do it
    - 25-50 >> User should not do it
    - 50-75 >> User should do it
    - 75-100 >> User should definitely do it

### 4.3.2 Decision Algorithm for Text Input

This algorithm differs from the category version, and follows the steps of:

- User types a text, describing the nature of their problem and submits.
- The Client side send only the following parameters to the NLP script at the server:
  - A text corpus, to be evaluated
  - The userId that belongs to the user
- The text is passed as parameter to pythonWatson function, which does the following:
  - Sends a curl request to IBMWatson NLP API including the text corpus and preferences.
  - Extracts the important keywords from the returned JSON object
  - Formats the keywords
  - Returns the result
- If the result is empty, the NLP function terminates and return the prompt for the user to input more detail
- If the result is non-empty, NLP script passes the same text corpus to csionLDA script, which does the following:
  - Loads the previously trained LDA model from the directory at server side.
    - The model was trained with the blog entry datas of more than 600,000 entries. The entries express a variety of emotions and mentioned various topics.
    - It is trained to generate 20 topics from the inputs.
  - Loads the gensim dictionary that is created from the training process, from the same directory.
  - Processes the input text for stemming and lemmatization.
  - Passes the result as parameter to the loaded dictionary objects doc2bow() function.
  - The dictionary creates a new 'bag of words'(BoW).
  - Finds the correspondents of the BoW vector in the LDA model and sortes it according to their relevance scores, so that the most relevant predicted topic for the text input is on top.
  - Stores the most relevant 50 keywords in the topic in a temporary variable and format them to get only the words and filter their scores.
  - Returns the most relevant 50 keywords to NLP script.
- The two keyword sets that are acquired from IBMWatson API and our LDA model are combined.
- User gets found by using getUserWithId function with parameter userId
- User's personality type's attributes get found by using getPersonalityType function with parameter personalityType string.

- Personality features are found by using the `getPersonalityType` function with the parameter `personalityType` string.
- For each of the acquired keywords in the combined list, each entry of every feature in the personality traits dictionary of the user is compared. Partial matches are accepted. Comparison are done according to:
  - If there is a match:
    - If the feature in the personality dictionary is positive( 'strengths','growths'):
      - Add the feature name to encourages list
    - If the feature in the personality dictionary is negative( weaknesses,'negativeCareer'):
      - Add the feature name to discourages list
  - Repeat this process until all of the keywords are iterated.
- Calculate the ratio of decision as:
 
$$100 * (\text{encouragesSize}) / (\text{encouragesSize} + \text{discouragesSize})$$
- Return encourages, discourages, ratio, and personalityType to frontend.
- When frontend receives this result, it displays a results screen the same as it is in the category score evaluation.

## 4.4. Database

We used DISC Personality types as default types in our application. We researched about the personality types in Crystal Knows Official Website [5]. After that, we cleaned personality type data and added new fields such as keywords which are the best for the inspecting personality. After researching the proper database cloud, we decided to use MongoDB Atlas as our cloud storage. Our data team was responsible to prepare data and push them to MongoDB Atlas in the decided standard format. Each member of data team prefer to write their own scripts to manage data and push them to database, which are reachable on our Github Page [6]. We decided that, for personality types, we stored:

```

1.  tends:Array
2.  strengths:Array
3.  weaknesses:Array
4.  growths:Array
5.  motivations:Array
6.  Stresses:Array
7.  positiveCareer:Array
8.  negativeCareer:Array
9.  jobs:Array
10. positiveFriendship:Array
11. negativeFriendship:Array
12. positiveRelationship:Array
13. negativeRelationship:Array
14. keywords:Array
15. type:"D"
```

```
16. name: "The Captain"
```

Also, our application includes a part which users can select their problems' categories, subcategories, and the problems. We decided to store problems in MongoDB Atlas, too. For Problems, we stored:

```
1. category: "Category_Name (Career/Relationship/Education) "  
2. subcategory: "Subcategory"  
3. problem: "Should I ...? (Users' specific problem related with categories and  
   subcategories) "  
4. personalQuestions: Array  
5. encourage: Array  
6. discourage: Array
```

Here, `personalQuestions` represents the question numbers that users will encounter if they select this problem. `encourages` and `discourages` are the specific keywords that we collected based on personality types' keywords.

For questions, we stored:

```
1. question: "Would you ...?"  
2. score: "2"  
3. questionId: "29"
```

Here, `questionId` represents the question number that we used to select whether this question will be shown to users for the given problem. `score` indicates the score if the user chooses Yes for this question. Based on this score, and matching encourage/discourage keyword of users' personality, we compute our answer for the given problem.

We store every relevant information regarding a user in the database as "user" dictionaries, with the following format:

```
1. name: "Name"  
2. surname: "Surname"  
3. username: "Username"  
4. password: "password"  
5. email: "mail@example.com"  
6. age: "25"  
7. __v: 0  
8. personalityType: "Dc"
```

Each user record's `objectId` in the Mongo serves as their `userId` in their sessions and their `personalityType` is used in the evaluation functions. When a user initiates a problem, a "campaign" is generated and stored in the database with the following format:

```
1. questionIds: Array  
2. answers: Array  
3. ownerId: "5ec918b5e148e5217a5d8e02"  
4. problem: Object  
5. category: "Career"  
6. __v: 0
```



In this collection `questionIds` store the indices of asked questions and `answers` are the user's answers to them. The `userId` of the user that generated this campaign is stored as `ownerId`, and the problem s/he selected is also available in the `problem` feature as a dictionary that holds relevant information regarding the problem.

Overall, this structure in the database enables us to operate and maintain the system we aimed to develop.

## 5. Testing Details

Throughout our implementation testing was one of the most important processes. We had to make sure that our application works optimally and it does not have any bugs or defects. We have created several different test cases and observed the behaviour of our application. We have adjusted our work according to feedback that we have gathered from test cases. We have used Selenium, Chrome DevTools for frontend and Mocha for backend testing [7,8,9]. We can divide our testing process in 4 parts.

### 5.1 Algorithm Testing

Our application is all about consistency and accuracy of an algorithm that we have implemented carefully and testing this algorithm with different parameters/situations was very critical. The behaviour of our algorithm against different cases has helped us to improve it. In order to test our algorithm we have needed to satisfy following criteria:

- Personality type of the user should be given
- The answers of the questions should follow the same structure
  - Yes means positive score which encourages user to act
  - No means negative score which discourages user to act
- Desire level should be indicated and it is between -10 to 10
  - Positive score means encouragement
  - Negative score means discouragement
- The score from answers and score from keyword match should weight equally

We have made sure that each criterion has been satisfied before testing our cases. We can summarize the test cases as following:

- The personality type with more matched encouraged keywords and high score which means great desire level with many “yes” answers. Should give more than 75% ratio as an output.

- The personality type with more matched encouraged keywords but low score which means less desire level with many “no” answers. Should give 25% to 50% ratio as an output.
- The personality type with more matched discouraged keywords and low score. Should give less than 25% ratio as an output.
- The personality type with more matched discouraged keywords but high score. Should give 25% to 50% ratio as an output.
- The personality type with equal number of matched encouraged and discouraged keywords but low score. Should give less than 25% ratio as an output.
- The personality type with equal number of matched encouraged and discouraged keywords but high score. Should give 50% to 75% ratio as an output.

These test cases were the initial ones for optimizing our algorithm. We have kept updating the algorithm until these test cases have passed for each different run. We have optimized the scores' weight and their calculation. When we have made sure that our test cases have stabilized, we have moved on to more specific situations such as dealing with limited information. We have needed to improve the algorithm for such cases but in order to avoid them we have tried to increase our keyword pool as large as possible. The test cases were following:

- Algorithm's behaviour against “problem” with limited encourage and discourage keywords
- Algorithm's behaviour against limited number of matched encourage or discourage keywords

Each test case has helped us to create a more accurate and consistent algorithm which is the most critical element of our implementation. When we had finished optimizing the algorithm, the application was close to finished.

## 5.2 Frontend/Client Testing

We have automated our frontend testing by using Selenium. Our user interface is not complicated but still interaction between different components and reaction of application is important. We can summarize our test cases as following:

- Sign Up as a new user
- Login as a user
- Personality test submission
- Functionality of each category buttons
- Functionality of each question buttons
- Filling the “yes” or “no” check boxes and setting the desire level
- See the profile page

- Update the user information
- See settings
- Functionality of “back” button for any situation

We also get use of Chrome DevTools on the developing phase of frontend for manual testing. By putting breakpoints on a line of JavaScript code, we were able to track our code line-by-line or instruction-by-instruction. We could also track the HTML DOM elements, and their CSS details at any moment. It has been very helpful for us to find and fix our bugs in the implementation of the user interface. Moreover, we were able to track sent requests and received responses (network actions) of our REST API there, so it also helped us to detect any problems related with client-server connection immediately.

These test cases helped us to check the implementation progress of the UI side of our application. For instance when “Filling the “yes” or “no” check boxes and setting the desire level” case has finally passed, we have started creating the connection between the algorithm and UI. Therefore, frontend testing has created a better working environment and organized the implementation process for us.

## 5.3 Backend/Server and Database Testing

We have created separate test cases for backend functions that communicate with databases by using Mocha. Our functions were considerably simple but still we have needed to make sure that every function works perfectly. Our tests can be summarized as following:

- Add user with defined parameters
  - username, password and email is required
- Get user with given userId
- Get user with given username and password
- Set personality type of a given userId
- Delete user with given userId and password
- Get problems for given category and subcategory
- Get questions for given category and questionIds
- Create Campaign with defined parameters
  - ownerId, problem, category, questionIds required
- Get campaign with given campaignId
- Get campaigns with given ownerId
- Update campaign with given campaignId

We have corrected our functions according to the results of our test cases. There were lots of exceptions and defects that we have handled based on output of these tests. We have managed to create a health connection with our database and our server.

## 5.4 NLP Testing

We have provided an open-ended way of providing the problem input by using Natural Language Processing (NLP). This was a secondary technique to enable a degree of flexibility in user input acquisition. Nevertheless, our implementation required a moderate degree of accuracy and high reliability. To test our implementation we needed the following criteria to be satisfied:

- The text input should be genuine. (No SQL Injection)
- The text input should be of adequate length
  - The input should hold enough specifiers to increase accuracy.
  - The input should be able to hold enough potential keywords.
- The text input should be provided in English.
- Emergency keyword generation should generate relevant keywords.

After each requirement was met, we tested our implementation by providing several example text corpuses. We have tested for cases including, but not limited to:

- Providing corpus of length smaller than 300, 400, 500 characters.
- Providing corpus with context of Relationship, Career and Education.
- Providing corpus with unique contexts.
- Providing corpus with poor choice of wording.

Overall in all of the test cases we aimed to acquire an adequate number of important aspects (decided as 10 for testing) from a text input and generate the missing amount by using synonyms of the words.

Each of our tests gave us a chance to work on our implementation and improve it, to provide a moderately accurate secondary use technique for our application.

## 6. Maintenance Plan and Details

In order to deploy and host our application we preferred to use Glitch platform, as they offer a flexible environment for us to develop the project cooperatively. Choosing to make a web application instead

of a mobile application also enabled us to be accessible in more platforms with little to no effort to update the project for any concern in an OS update. Glitch was also free to use, so it proved to be a good origin point for us to start a project. However, with the addition of the LDA model to our project we needed to store the trained model on the server since it was not feasible for the server to download it every time the evaluation methods were invoked. In order to gain enough storage for our model we had to go boost our project with a monthly fee. Glitch also keeps our application awake at all times, which was a major upside to choosing it over hosting our app ourselves, being restricted to certain time constraints.

For our database we preferred to go along with MongoDB, which provides fairly simple user interfaces for the developers and secure access. One upside of choosing Mongo was it dedicating 512 MB of space, which after deducing the space it takes to hold the personality, problem and question collections leaves enough space for approximately 1000 users to actively use our application. Upon achieving a full user load we are able to upgrade to a larger plan with a minimal monthly fee, which provides a good degree of scalability for our application in its deployment stage.

Currently our MongoDB and Glitch deployments are able to operate with ease, since our application has 2 GB of memory available and Mongo can handle our database interactions. Upon facing an increase in the number of users and possibly user enquiries we may need to move our project to another platform, rent a dedicated server or distribute our project amongst several deployments, rather than sticking with Glitch platform alone, in order to match the demand.

## 7. Other Project Elements

In this part of our final report we talked about consideration of various factors, ethics and professional responsibilities judgements, impacts to various contexts, teamwork and peer contribution, project plan observed and objectives met, new knowledge acquired and learning strategies used in the respective order.

### 7.1.Consideration of Various Factors

Table 1: Factors that can affect analysis and design.

	Effect level	Effect
Public health	8/10	If the algorithm is not designed properly, regardless of the filters, or any filters are forgotten or crashed, it may threaten the public health; like suicides, homicides etc. Hence, filters are designed faultless.
Public safety	8/10	As mentioned above, if the decision mechanism crashes, it may make their users threaten public safety incase of violating society. To avoid such cases, databases are limited with the questions and answers by developers.
Public welfare	6/10	Increase in public welfare may increase the use of the application; if public welfare is well, the rate of buying and downloading the application will be increased. Moreover, maybe we, the developers are able to extend the program with more advanced tools. However, it does not have a significant importance in comparison with other factors.
Global factors	8/10	Unexpected changes in global factors may affect application in a bad way, since its learning and prediction algorithm may be needed to be renewed. To exemplify, in these days regarding Corona Outbreak,

		personality test results might be affected. Because, users can be in a depressive mood.
Cultural factors	10/10	If the application is not accepted by a specific culture, this will damage its market value. Decisions that may include sensitive ideas might not be displayed. To exemplify, programs should not give answers to specific topics which include region, politics, etc. To avoid such cases, databases are limited with the questions and answers by developers.
Social factors	10/10	The answering algorithm should be designed according to various social factors. Given answers should not violate any social relations. Moreover, its answering and prediction algorithm may differ in different regions' or populations' different social factors. System should be able to understand the user's social relations, situations etc then give answers by taking these into consideration. If not, the program will not be preferred by the users. To avoid such cases, databases are limited with the questions and answers by developers.

## 7.2.Ethics and Professional Responsibilities

- All collected sensitive user data from questionnaires and personality tests are used and stored according to General Data Protection Regulation (GDPR) by following their guidelines **[10]**.
- All collected sensitive user data are transferred using encrypted protocols like HTTPS.
- User passwords are stored as salted hash.
- All sensitive user data should be encrypted with a symmetric key.
- We are transparent about data collection, data processing and we inform our users about it.
- Consent of users is taken before processing their data to improve our application.
- To make sure that users' data are safe, we take database backups frequently and store those backups in different locations.
- We respect our users' right to be forgotten and we are able to clear their sensitive data without any problem.

- We created a clear Terms & Conditions form and made sure that users read and accept it before using our application.

### 7.3.Judgements and Impacts to Various Contexts

Table 2: Judgements and Impacts

Judgement Description	Protecting user data	
	Impact Level	Impact Description
Impact in Global Context	10/10	Data protection is a crucial topic for application users these days. Hence, we take consent of users to take their data and ensure that we do not share it with 3rd party applications.
Impact in Economic Context	10/10	If we violate any term of conditions about our data protection, we may be able to pay users for our damage.
Impact in Environmental Context	0/10	This judgement is not relevant with our application.
Impact in Societal Context	6/10	Our application provides alternative solutions based on users' personality yet, users are free to make their decision. The answers and questions are stored in our database and will not be shared with any other resource.

Judgement Description	Language used in questions and answers	
	Impact Level	Impact Description
Impact in Global Context	5/10	Applications which include offensive ideas are not recommended in the global area yet, they are not permitted too. Hence, impact on global effect is not very useful for this judgement.
Impact in Economic Context	10/10	If we do any disrespectful activity or promote/advise it, we



		may be able to pay for our damage. Hence, our questions and answers are designed in an unchangeable format.
Impact in Environmental Context	0/10	This judgement is not relevant with our application.
Impact in Societal Context	10/10	Since our application aims to help its users' with their problems, we need to be sure that our context should not give any damage to users and their values. Hence, we decided to write our questions and answers based on these considerations.

Judgement Description	Competition in Global Market	
	Impact Level	Impact Description
Impact in Global Context	10/10	As we observed, personality tests gained popularity in 2019-2020. Hence, we decided our application based on society's interests. We used Crystal Knows as our resource, which is also a very popular personality test resource these days.
Impact in Economic Context	10/10	According to our research, there is no application to give people decisions about their personality. We believe that Csion is an innovative application, hence in economic context, it can gain popularity and provide income.
Impact in Environmental Context	0/10	This judgement is not relevant with our application.
Impact in Societal Context	5/10	For Competition Judgement, there is no significant Societal Context that we highlighted.

Judgement Description	Scaling after Deployment	
	Impact Level	Impact Description
Impact in Global Context	5/10	In Global Context, our application is able to provide service to several people in distress and will be able to serve many more with scaling.
Impact in Economic Context	0/10	This judgement is not relevant with our application.
Impact in Environmental Context	0/10	This judgement is not relevant with our application.
Impact in Societal Context	7/10	By combining a certified personality test with categorical problem evaluation, we have provided an advice system for those people in need.

## 7.4 Teamwork and Peer Contribution

For the continuous workflow and fair leadership of our project, we divided our team to apply different parts of the project. For each part, a different member of the team lead. In that case, the members who have more knowledge on specific areas, shared their knowledge with other members and as a result of that, we could reach mutual knowledge and common implementation decisions. During the project, the following members worked on the given fields:

**Nursena Kurubas:** Design and implementation of User Interface of the Csion

For the implementation details of Nursena Kurubas: <https://github.com/aeyc/cSION/tree/nursena>

**İsmail Yavuzselim Tasci:** Design and implementation of the server of the Csion

For the implementation details of İsmail Yavuzselim Tasci:

<https://github.com/aeyc/cSION/tree/yavuz>

**Mehmet Selim Özcan:** Design and implementation of personality and questions database, NLP

For the implementation details of Mehmet Selim Özcan:

<https://github.com/aeyc/cSION/tree/selim>

**Mehmet Sanisoglu:** Design and implementation of personality and questions database, NLP

For the implementation details of Mehmet Sanisoglu:

<https://github.com/aeyc/cSION/tree/mehmetDev>

**Ayca Begum Tascioglu:** Design and implementation of personality and questions database, NLP

For the implementation details of Ayca Begum Tascioglu:

<https://github.com/aeYC/csion/tree/ay%C3%A7a>

For our integrated version of our program (with the algorithms) please check:

- <https://glitch.com/edit/#!/csion>
- <https://github.com/aeYC/csion>

## 7.5 Project Plan Observed and Objectives Met

Looking back at the project plan that we created back at the Analysis Report, we can say that we accomplished to complete all work packages that we planned to do to create the base of our project. However, through the implementation stage, we made some changes in people involved to each package. The plan we did at the beginning was that all of us will contribute to each work package, and we will work altogether by meeting frequently. On the other hand, this semester we had some difficulties meeting and working together due to COVID-19 outbreak, so we decided to divide the work packages between us as much as we can to decrease our dependence on each other.

Although we tried to follow our project plan and managed to finish other works at the times we planned, there were some delays occurred at the development of UI, NLP and server part of the project. Nevertheless, we ended up with a basic version of our application, involving all things that we think as musts. At the end, we can say that our objectives are met, and we have the first version of our application project as the deliverable. In fact, we were thinking to develop this project only as a mobile application, now we also have its web-page version for the ones who prefer to use it on desktops and laptops.

## 7.6 New Knowledge Acquired and Learning Strategies Used

So far, we have done many online researches about the personality tests to choose the one that is most useful for us, and at the end, we decided that Crystal Knows Personality Assessment that is based on 16 personalities is the best that we can use since it also provides a full test API for developers. We also investigated how to customize and manipulate Chatbot APIs and how to use the NLP technology to understand and process users' problems. We also arranged several meetings with our innovation expert to discuss the details and feasibility of our application, we had his ideas about the Chatbot implementation, and we created our plan accordingly. While making decisions about our

implementation, we also share our knowledge and experience about different areas and make our design choices by considering the ideas and skills of our peers.

We watched some crash-course videos and made use of online resources for the technologies that we did not have any experience from in the past. After a broad research on the Internet and discussing with some people with experience, we decided to use IBM Watson technology for the NLP part of the project and we had hands-on experience to learn it since we had a limited time and we thought it would be a lot faster this way [11]. At this part, we learned more about how natural language processing works, as we should have. Most importantly, we solidify our skills on the coding languages that we used in our project, which are JavaScript and Python, and we practiced them by doing exercises on coding platforms. Through the coding phase, we excessively get use of consultant web sites, like StackOverflow [12]. For database implementation, we chose to use MongoDB, we did lots of online research and interviewed with experts before choosing it, and then we decided that it is a convenient technology for our project. In fact, another reason for us to choose this technology is that we wanted to learn and experience a database technology different from we are used to. Finally, although the ones who deal with this part know ML technology, we used a new model named LDA for our NLP section, which we again learned through the related websites [13].

## 8. Conclusion and Future Work

In conclusion, in our senior project we implemented nearly all of the features that we designed in the previous phases of the project and we are really happy with the end result. However, we are aware that we can improve our idea and project greatly by improving some aspects of it. We can list those aspects as follows:

- **Improving User Experience:** About this aspect of our project, we did what we can in this limited time. We tried to design our frontend before implementing and we actually tried to make decisions according to general UI/UX rules. However we are aware that we are not professional designers and if we have a designer, we believe that we can improve our user experience greatly. Since user experience is very important for applications nowadays, our project will benefit greatly from improving it and this will increase our applications chance to have a success when we publish it.
- **Improving Decision Making Algorithm:** Right now we have a good algorithm to make suggestions to users but it is nowhere near perfect. It needs fine tuning with the help of a psychology expert. Since we have different weights for different questions and for different personalities, our algorithm is actually really open to improvements. Once we have a chance to work with experts in this field, we believe that we can tweak individual weights to have

more appropriate algorithms and our result will become much more accurate which will result in a more satisfactory user experience and eventually success of our application.

- **Adding Deep Learning:** One of the biggest improvements that we are planning is adding deep learning to our application. Since we are getting feedback from users for each of their problems, we are planning to use that data to train our deep learning model to understand how each personality type needs to act in certain situations. We believe that this has huge potential if we can collect enough data because our personality test is very accurate and if we can understand the previous data about a personality type, we can actually make very accurate suggestions for that personality type. Since our data will grow constantly, we will feed that data to our algorithm constantly as well. So we expect to have a deep learning model that will improve in time, but eventually it will be good enough to make really accurate suggestions to our users. Again like improving our decision making algorithm, this improvement will also increase our reliability about our suggestions which will lead to better user experience and eventually lead us to success.
- **Providing Statistics For Users:** Adding deep learning to make use of our data is one way to use our data for our users interest. Another way is, we can actually publish our raw data in human readable form with the help of charts and figures. By doing that, we will enable the possibility of users' own interpretation of our data. Basically our users will be able to see how other people act in case of similar problems and how satisfied they are. Of course, it will be categorized by personalities and our users will be able to see the chart that belongs to their own personality. And of course, there will be no identifier associated with the data so no one will be able to see how specific individuals acted previously. So all the data will be anonymous. We believe that this feature will become a valuable asset for our application and once our data becomes large, it will become a very good helper for our users to make their decisions.
- **Adding New Categories:** For now, we are able to make suggestions about 3 general topics and in those topics we have limited amounts of subtopics. We know that, if we want to expand our audience as much as we can, we should definitely increase our topics. Because every user will be interested in different topics and we cannot say that our topics will span all of the users' interests. However, for a good start we tried to choose topics that are very common across people. Moreover to this point, while developing our application, we always had the idea of expanding our topics, so we tried to make everything as scalable as possible. Because of that right now, it is very easy to add new topics and subtopics to our already existing topics. In the future, by adding more topics, we will expand the coverage of our application.

- **Marketing:** Without marketing, our application cannot create its own audience because people will not know about it and even if we had a very good application it cannot achieve success. Because of that reason, after we are satisfied with the situation of the application, we are planning to start marketing immediately. Actually we are not planning to run big marketing campaigns but we will start spreading the word about our application starting with our families and friends. After that, we are planning to run Google Ads campaigns to reach more audiences, and actually this part is very important because we have a very limited marketing budget and we want to reach as many people as we can. After we get our first users, we believe that our application will grab the attention of our users and our users start to recommend it to their friends and families. Hence we believe that we can start creating a good number of users with a small-sized marketing campaign.

## 9. Appendix - User Manual

Here you can find our user manual with real screenshots of every page from our live app. Bottom of every page you will find the various interactions that page allows. Also you will find how to install our mobile application version.

### 9.1 Csion Application Usage

Here you will find usage instructions of our application.

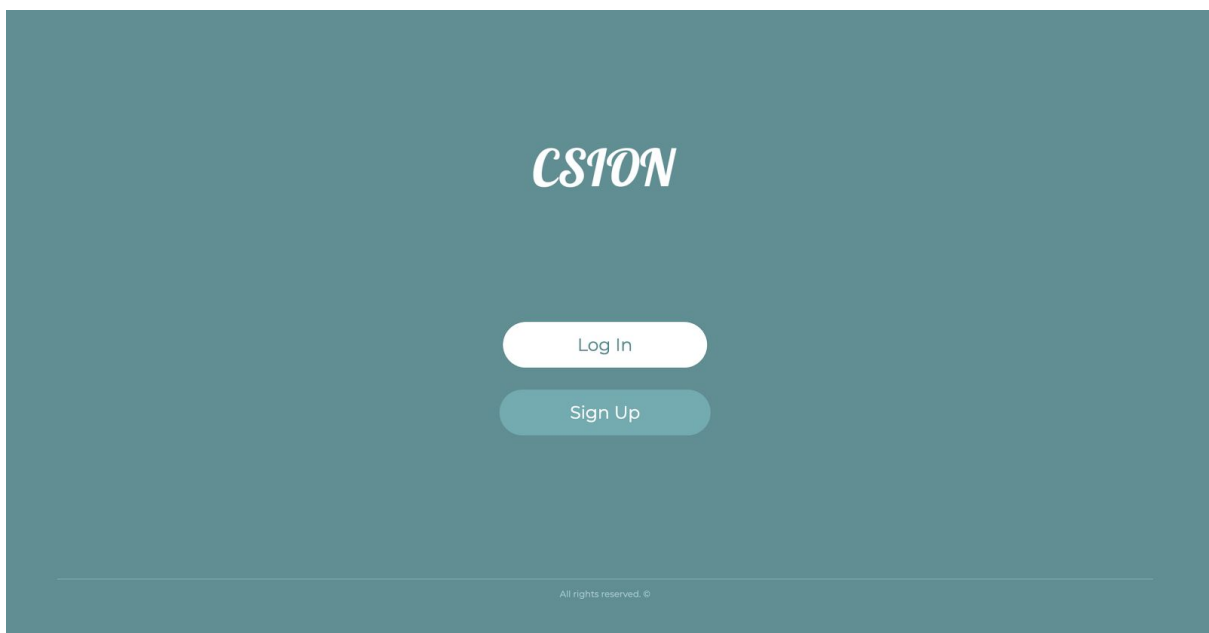


Figure 10: Landing page

This is the landing page of our application. From here by clicking the login button, users can access login form or by clicking sign up button users can access sign up form.

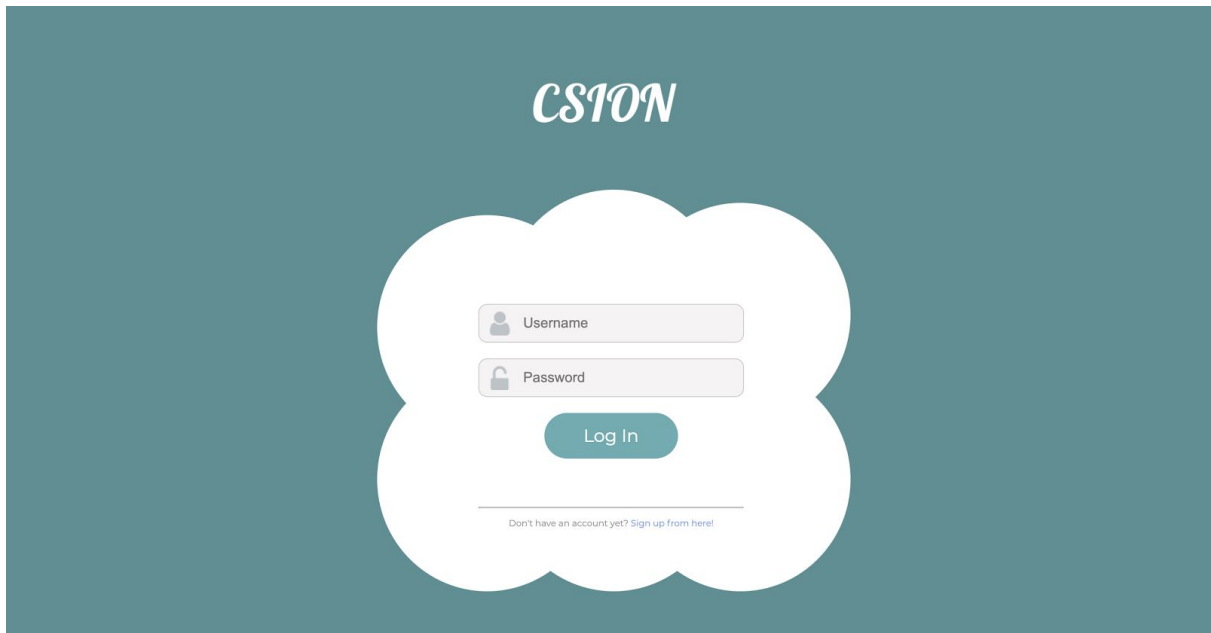
The image shows a login form on a teal background. At the top center is the logo "CS10N" in a white, stylized font. Below the logo is a white cloud-shaped container. Inside the cloud, there are two input fields: "Username" with a user icon and "Password" with a lock icon. Below these fields is a teal "Log In" button. At the bottom of the cloud, there is a link that says "Don't have an account yet? Sign up from here!".

Figure 11: Login form

This is our login form. From here, users can fill out their login information and by clicking log in button they can log in to our application. If there is an error about authentication information that the user entered, form fields will turn to red and notify the user about it. If a user did not sign up yet, by clicking Sign up from here button, they can access sign up form. After logging in users will be redirect to either the personality test page or homepage depending if they took the test already or not.

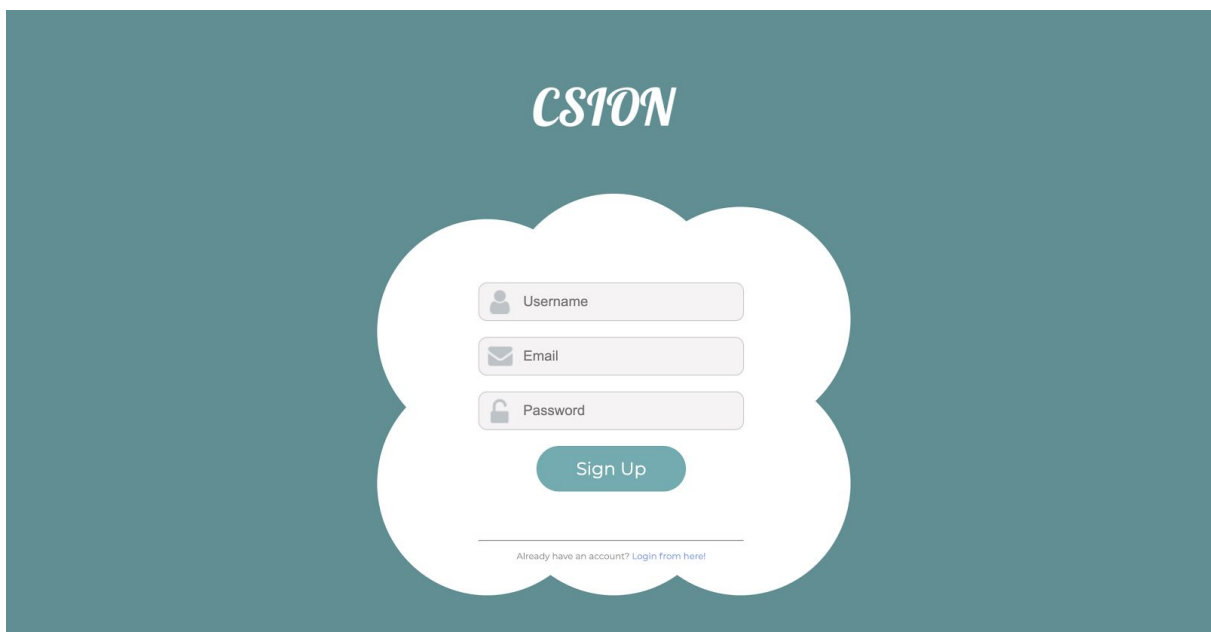
The image shows a signup form on a teal background. At the top center is the logo "CS10N" in a white, stylized font. Below the logo is a white cloud-shaped container. Inside the cloud, there are three input fields: "Username" with a user icon, "Email" with an envelope icon, and "Password" with a lock icon. Below these fields is a teal "Sign Up" button. At the bottom of the cloud, there is a link that says "Already have an account? Login from here!".

Figure 12: Signup form

This is our sign up form. With using this from users can fill out their information required to sign up, username email, password, and by clicking the sign up button they can sign up to our application. After signing up they will be logged in automatically and they will be redirected to our test page to take our personality test. If there is an error with the information they enter in this form, they will be notified also. If a user has already signed up and he or she wants to log in, they can also access login form by clicking Login from here button.

Select the word that most describes you and the word that least describes you.

I am...	Restrained ⓘ	Decisive ⓘ	Chatty ⓘ	Traditional ⓘ
Most	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Least	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Powered By: Crystal ⓘ

NEXT

Figure 13: Personality Test Page

This is our personality test page. From this page users can answer the test questions by clicking the radio buttons. By clicking next they can move to the next questions. After the test finishes users will see the result result page.

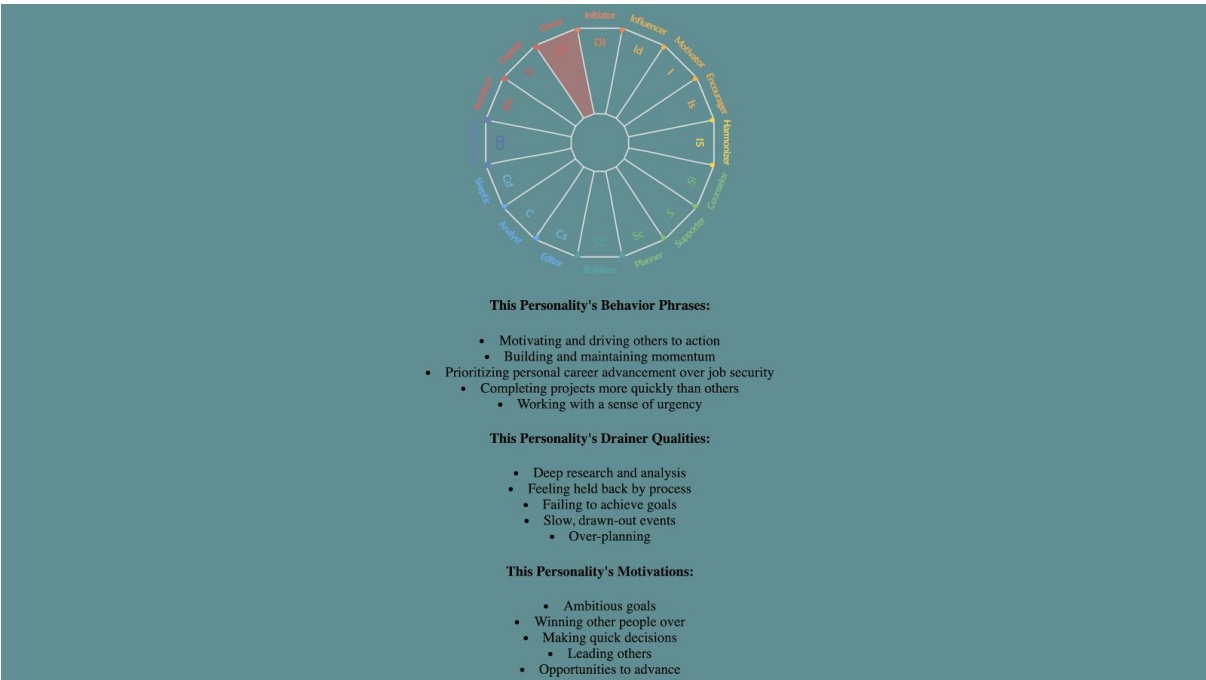


Figure 14: Personality Result Page

This is our personality test result page. From this page users can easily see their result, personality type and many related information about their personality. After they view this page, they can continue to the home page by clicking the Continue button at the end of the page.



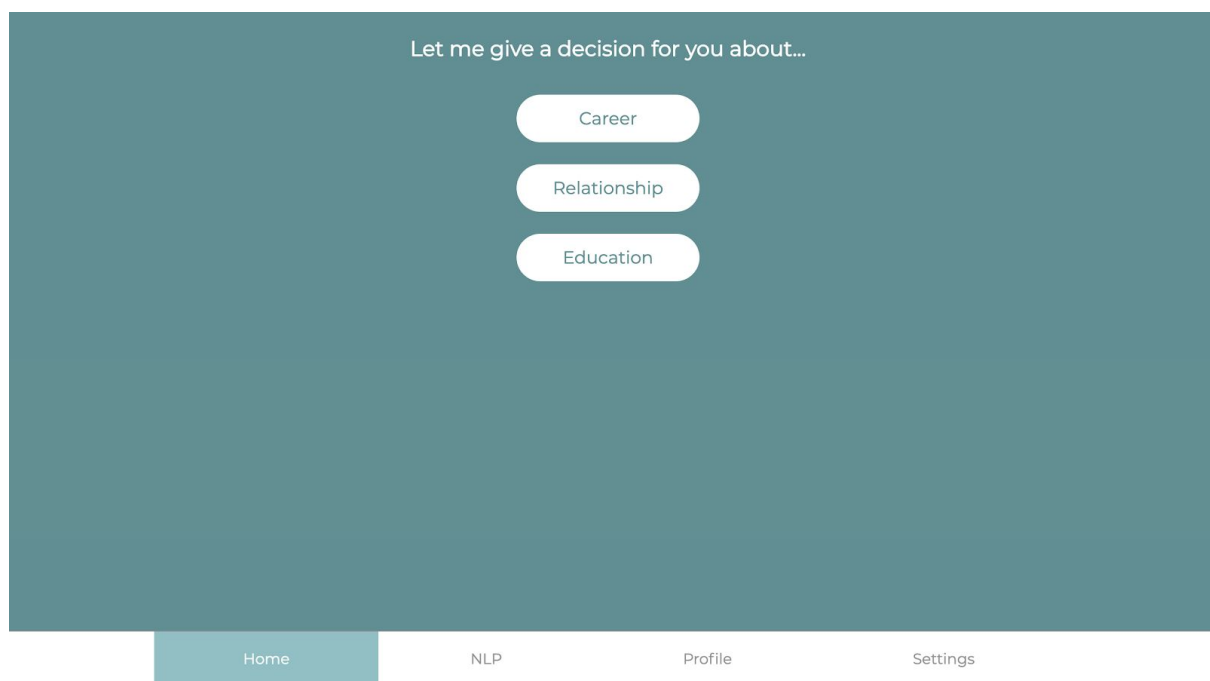


Figure 15: Home page

This is our home page. From this page users can choose the problem topic that they want to get a suggestion. Also from now on every one of our pages will include a navigation bar at the very end of the page. From this navigation bar users can navigate to other pages by clicking respective buttons. For example by clicking the NLP page a user can navigate to the NLP page.

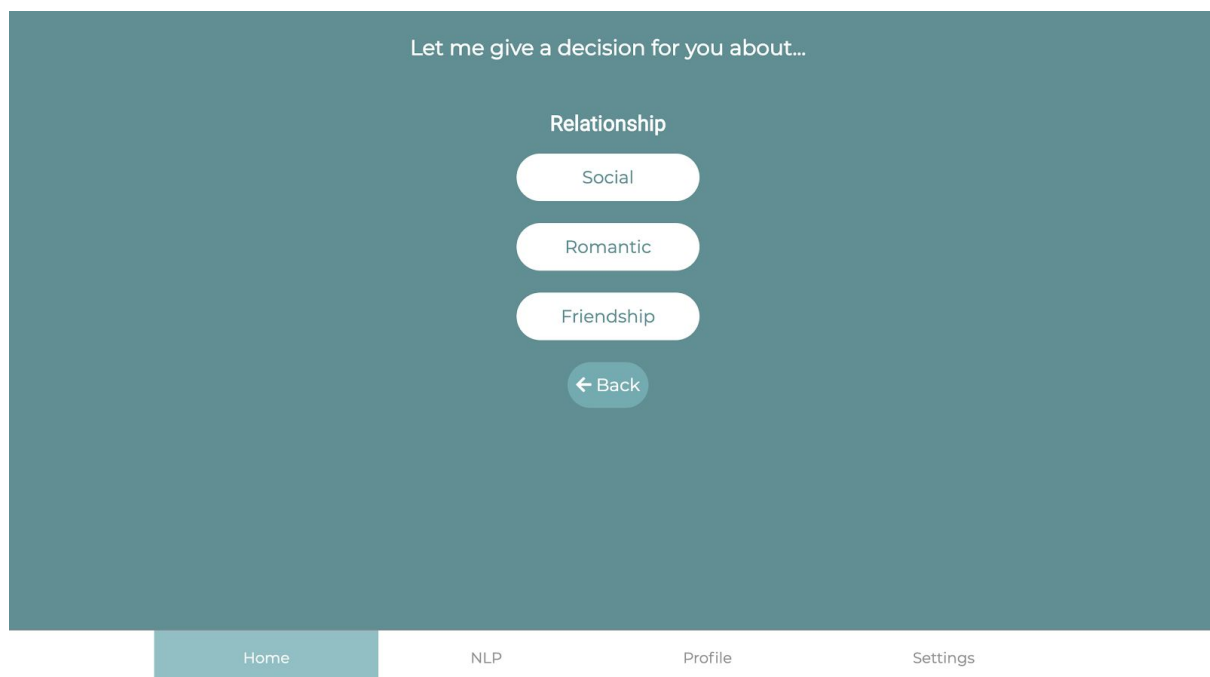


Figure 16: Subcategory page

This is our subcategory page. From this page users can see the subcategories that is related to the category that they choose in the home page. By clicking one of the subcategory buttons users can see the problems page or by clicking Back button users can navigate back to the category selection page.

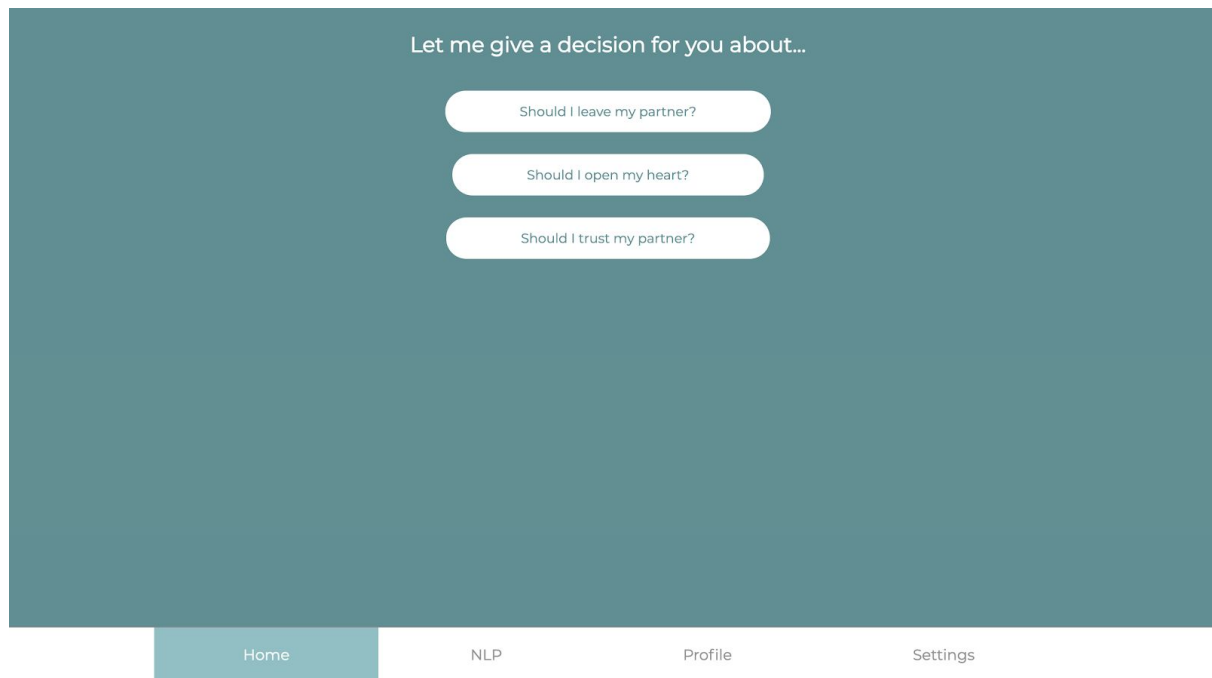


Figure 17: Problem page

This is our problem page. From here a user can choose a specific problem to see the questions specific to that problem. This action can be performed by clicking problem buttons.

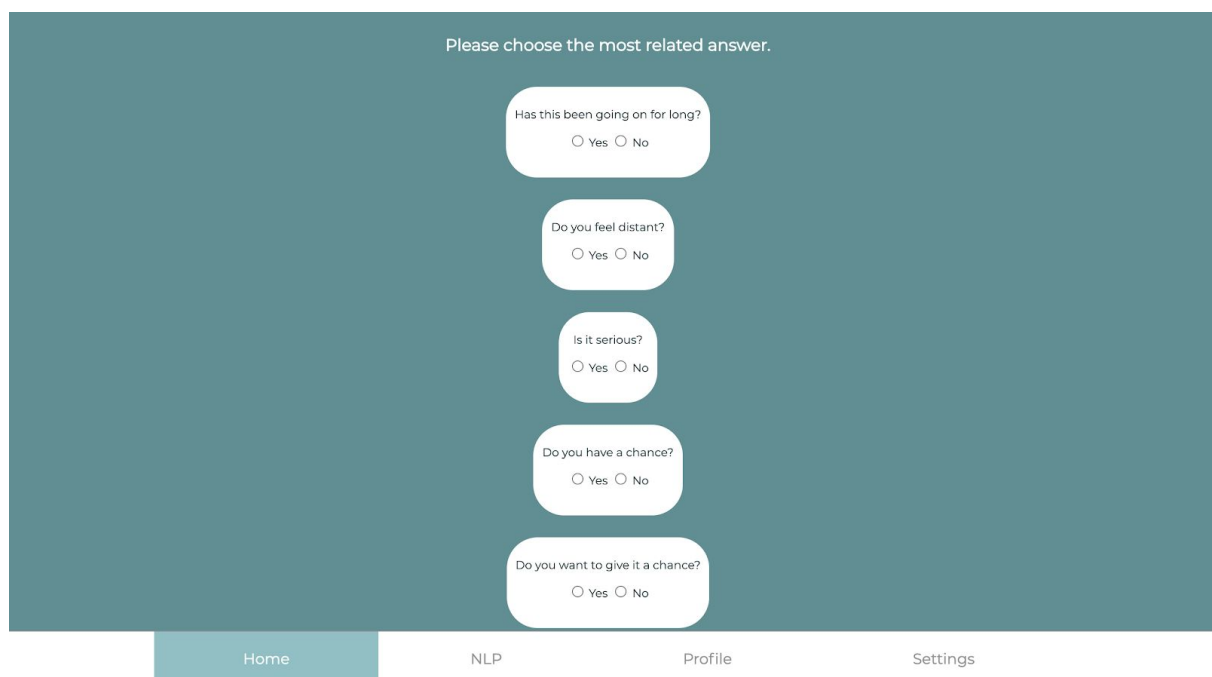


Figure 18: Questions page

This is our questions page. From this page, a user can answer our question about the problem. Since our algorithm works by collecting the data about the problem this question has to be answered by the user. These questions all consist of yes and no questions and users will be able to answer the questions by clicking radio buttons that reside at the bottom of every question. At the end of the questions there is a Submit button. By clicking that button users can submit their answers to our servers and they will be redirected to a suggestion page.

The screenshot displays a web interface for a psychological assessment. It features a dark teal background with light teal rounded rectangular boxes for text input. The content is organized into five main sections, each with a title and a list of questions or attributes. At the bottom, there is a navigation bar with four buttons: 'Home', 'NLP', 'Profile', and 'Settings'. The 'Home' button is highlighted with a lighter teal background.

**Your Problem:**  
Should I open my heart?

**Your Answers:**  
Has this been going on for long? : Yes  
Do you feel distant? : No  
Is it serious? : Yes  
Do you have a chance? : Yes  
Do you want to give it a chance? : Yes  
Do you know him / her? : Yes  
Isn't it worth trying? : Yes  
Do you want things to change? : Yes

**Your Personality Type:**  
DI

**Supportive Personality Attributes:**  
Brave - Convincing - Straightforward - DecisionMaker

**Discouraging Personality Attributes:**  
Forceful - Aggressive - Inconsistent

Home    NLP    Profile    Settings

Figure 19: Output page

This is our suggestion page. From this page a user can see our detailed analysis and suggestion about his or her problem. After viewing our suggestions a user can click to the Back button that is placed at the end of the page to go back to the home page.

Please describe your situation...

After your explanation ask your question in the form of "Should I do this?"

My girlfriend is extremely close with her guy friend. She hugs and kisses him and sits on his lap. Everyone thinks THEY are the couple. I told her about it and she got angry and broke up with me. Should I try to get her back or is she not worth it?

Submit

Home NLP Profile Settings

Figure 20: NLP page

This is our NLP page. In this page users can type their problem to the textbox. After typing, by clicking the Submit button they can submit their writings to our servers for processing. After clicking Submit button users will be redirected to the suggestion page.

Here is your result!

Your Personality Type:  
Dc

Supportive Personality Attributes:  
Rapid Feedback - Feedback - Direct Communication - Clear Communication - Communication - Formal Communication - Realistic - Careful - Factual - Practical

Discouraging Personality Attributes:  
Communication - Delivering Criticism - Emphaty - Little Responsibility - Changing

Our Advice:  
In the light of the data we gathered, we believe that it would be very bad for you to perform this action.

← Back

Home NLP Profile Settings

Figure 21: NLP Output page

This is the suggestion page that users will see after submitting their writings. It is very similar to what they see when they use our question - answer based system.

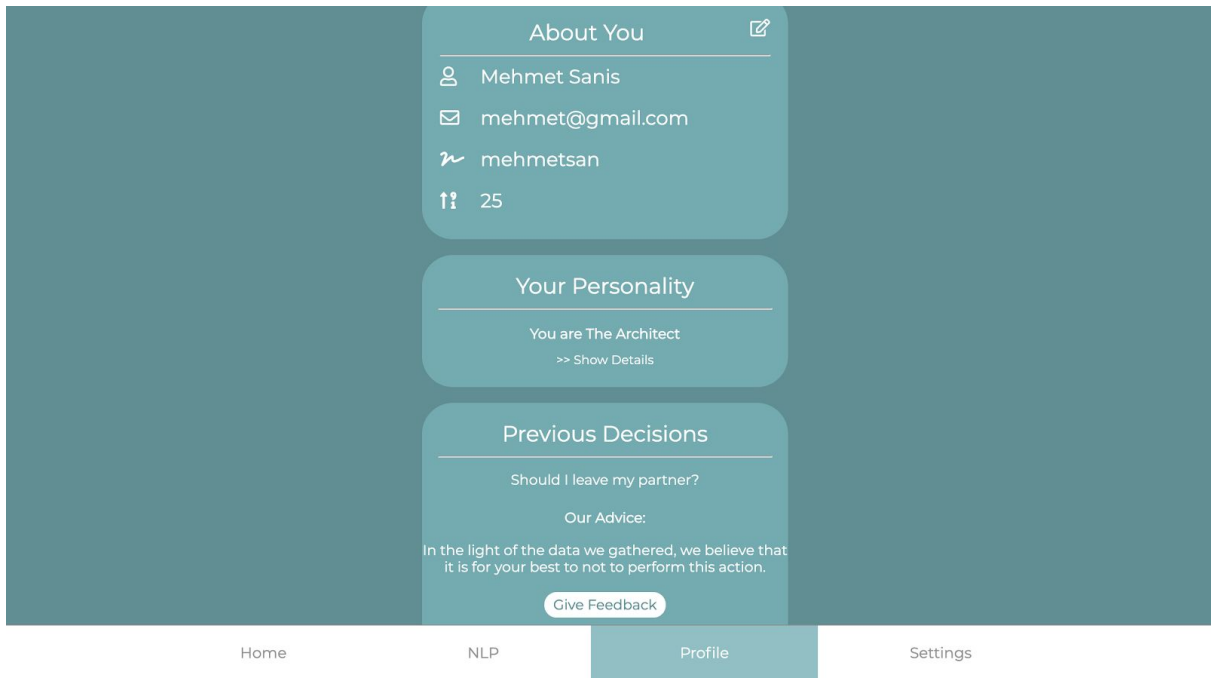


Figure 22: Profile page

This is the profile page of the user. From this page a user can change their name, username, email, or age. Also they can view their personality. If a user click Show Details button, they will also see a very detailed explanation of their personality. Also users can see their previous questions to our system and our answers. They can also Give Feedback about our suggestion by clicking the Give Feedback button.

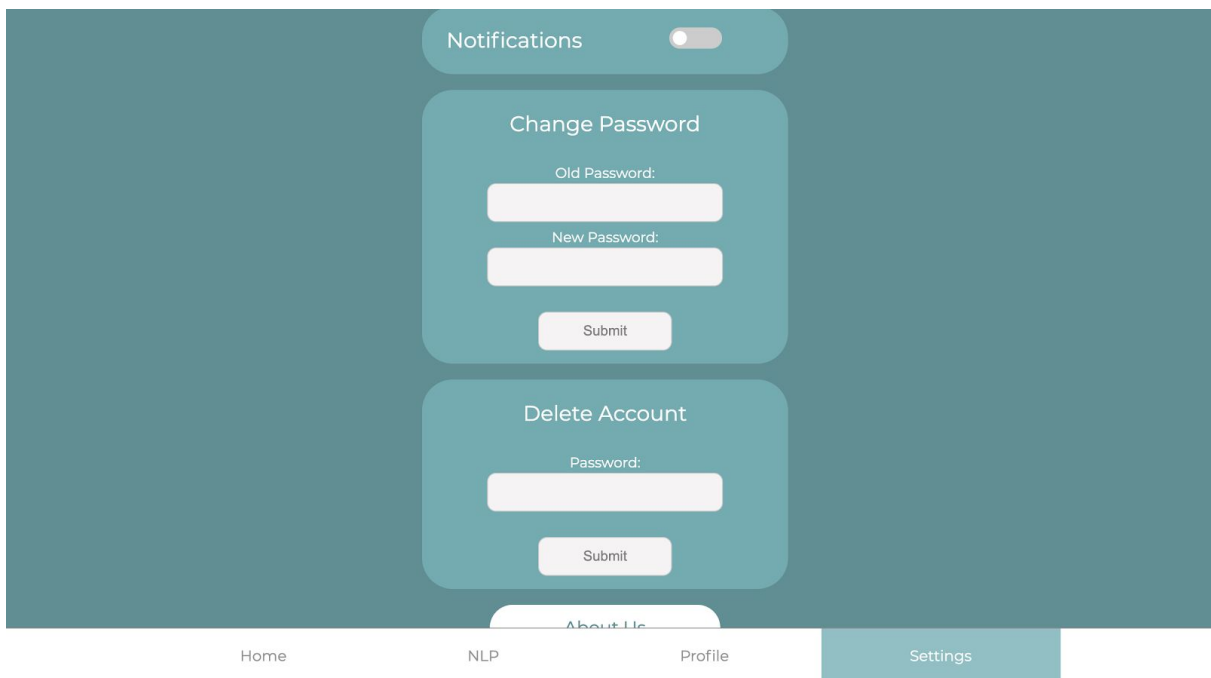


Figure 23: Settings page

This is our settings page. From this page users can control the settings of their account. They can turn on and off their notifications by clicking the switch next to Notification text. Also they can use Change Password form to change their passwords to a new one. For this they need to enter their old password and a new password and they must click the Submit button at the bottom of the form. Also they can delete their account from this page by entering their password to the corresponding text field and by clicking the corresponding Submit button. We have two more buttons at the end of this page which are Log Out and About Us buttons. By clicking About Us button, users can get information about us - who developed this application - and by clicking Log Out button they can log out from their accounts.

## 9.2 Csion Mobile Application Installation

In this section we have explained how to install Csion as a mobile application.

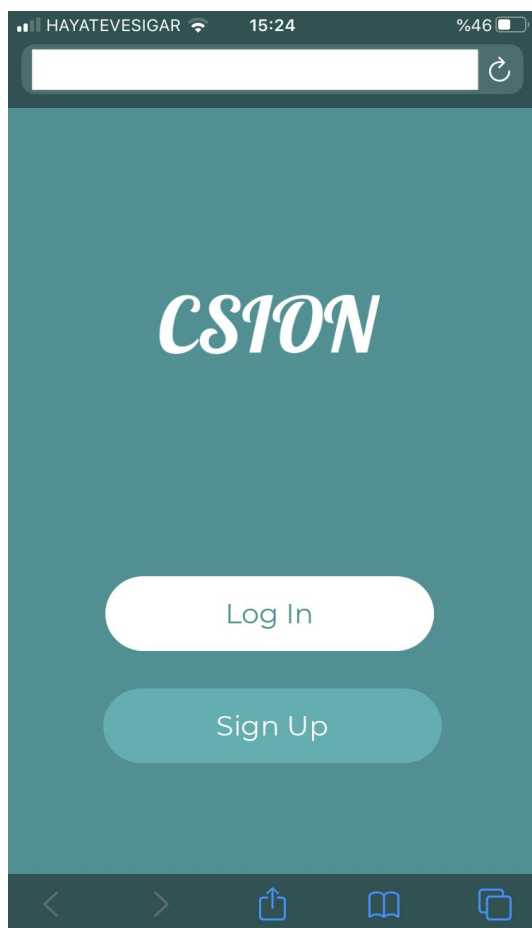


Figure 24: iOS landing page

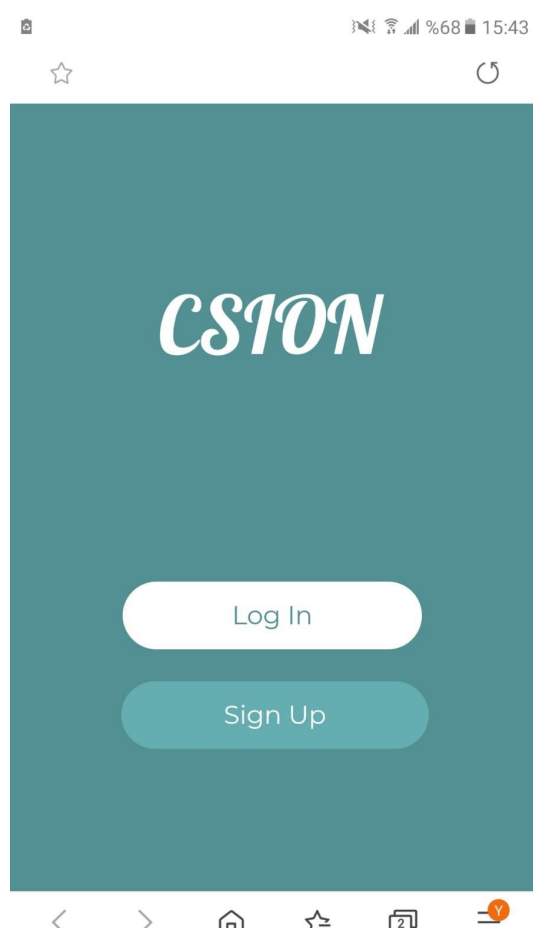
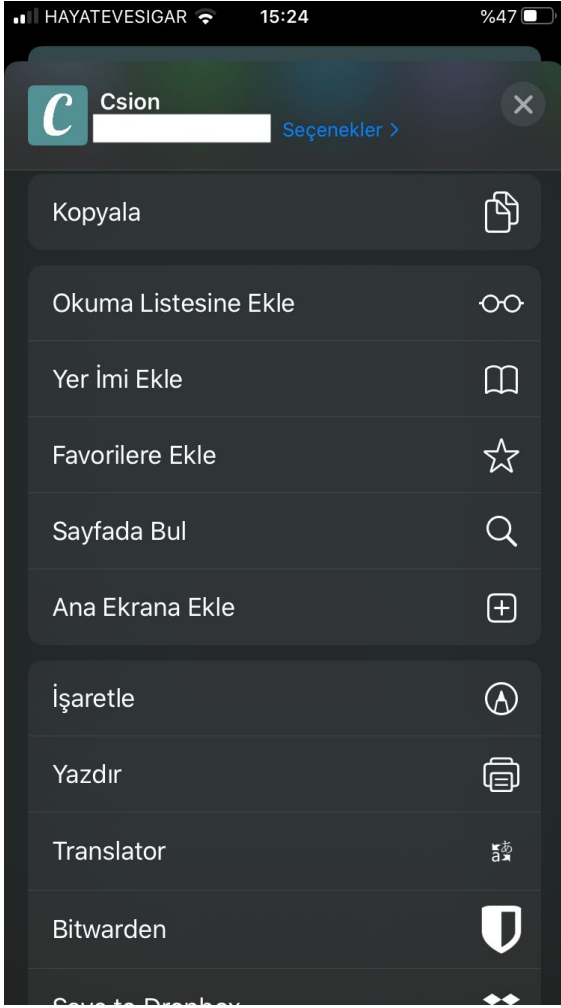
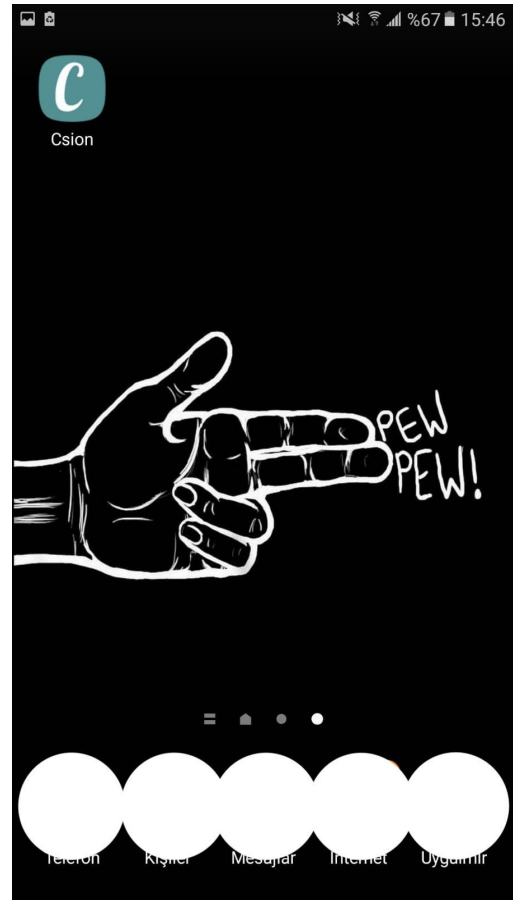
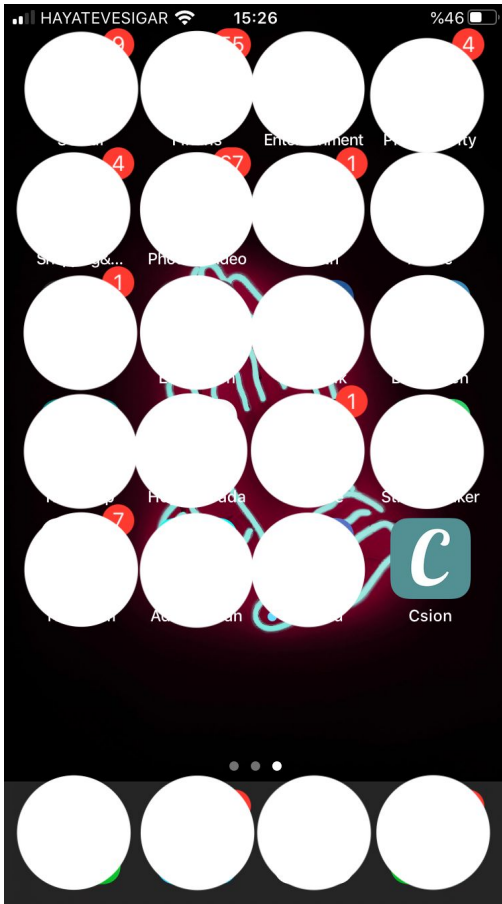


Figure 25: Android landing page

First of all you need to go to our applications website. For iOS, you need to click the book icon in the toolbar below. For Android, You need to lick the rightmost icon in the toolbar.



For both OS, after last step these menus will popup. From these menus click “Add To Main Menu” button and a new popup will open. Then click “Add” button on the popup.



Here is your installed application. Enjoy!!



# References

- [1] "The purpose of the Myers-Briggs Type Indicator® (MBTI®) ," The Myers & Briggs Foundation - MBTI® Basics. [Online]. Available: <https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1>. [Accessed: 27-May-2020].
- [2] "Mongoose," Mongoose ODM v5.9.16. [Online]. Available: <https://mongoosejs.com/>. [Accessed: 27-May-2020].
- [3] "The Friendly Community Where Everyone Builds the Web." Glitch, [glitch.com/](https://glitch.com/). [Accessed: 27-May-2020].
- [4] Inconshreveable. "Public URLs for Exposing Your Local Web Server." Ngrok, [ngrok.com/](https://ngrok.com/). [Accessed: 27-May-2020].
- [5] "Developers." Help Center, [docs.crystalknows.com/developers](https://docs.crystalknows.com/developers). [Accessed: 27-May-2020].
- [6] Aeyc, "aeyc/csion," *GitHub*, 27-May-2020. [Online]. Available: <https://github.com/aeyc/csion>. [Accessed: 27-May-2020].
- [7] "Selenium automates browsers. That's it!," SeleniumHQ Browser Automation. [Online]. Available: <https://www.selenium.dev/>. [Accessed: 27-May-2020].
- [8] "the fun, simple, flexible JavaScript test framework," Mocha. [Online]. Available: <https://mochajs.org/>. [Accessed: 27-May-2020].
- [9] "Chrome DevTools & Tools for Web Developers & Google Developers," Google. [Online]. Available: <https://developers.google.com/web/tools/chrome-devtools>. [Accessed: 27-May-2020].
- [10] "General Data Protection Regulation (GDPR) Compliance Guidelines." GDPR.eu. [Online]. Available: <https://gdpr.eu/>. [Accessed: 26-May-2020].
- [11] "IBM Watson." IBM, [www.ibm.com/tr-tr/watson](https://www.ibm.com/tr-tr/watson). [Accessed: 26-May-2020].

[12] “Where Developers Learn, Share, & Build Careers.” Stack Overflow, [stackoverflow.com/](https://stackoverflow.com/).  
[Accessed: 26-May-2020].

[13] Li, Susan. “Topic Modeling and Latent Dirichlet Allocation (LDA) in Python.” *Medium*,  
Towards Data Science, 1 June 2018,  
[towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24](https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24). [Accessed: 26-May-2020].