Information System Report

25^{th} Group

April 12, 2018

Contents

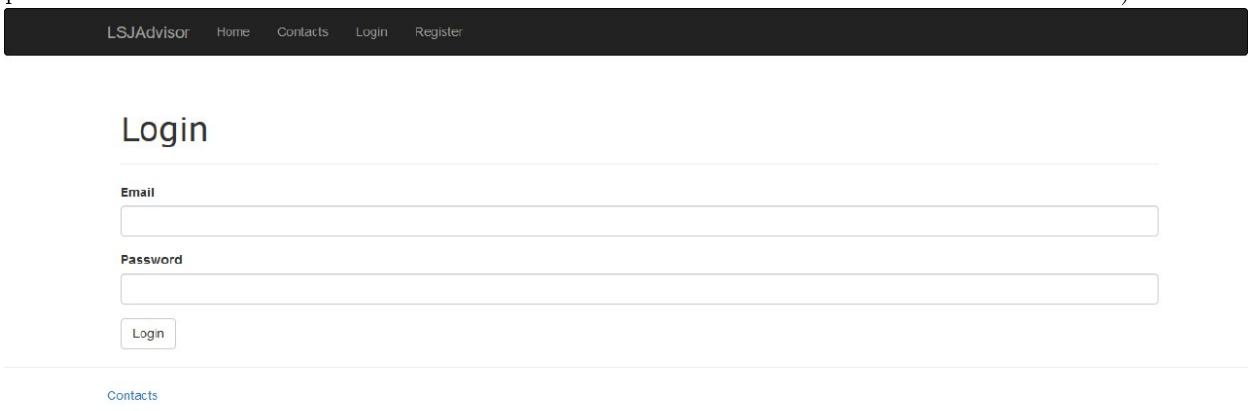
1 Introduction

1.1 LSJ Advisor

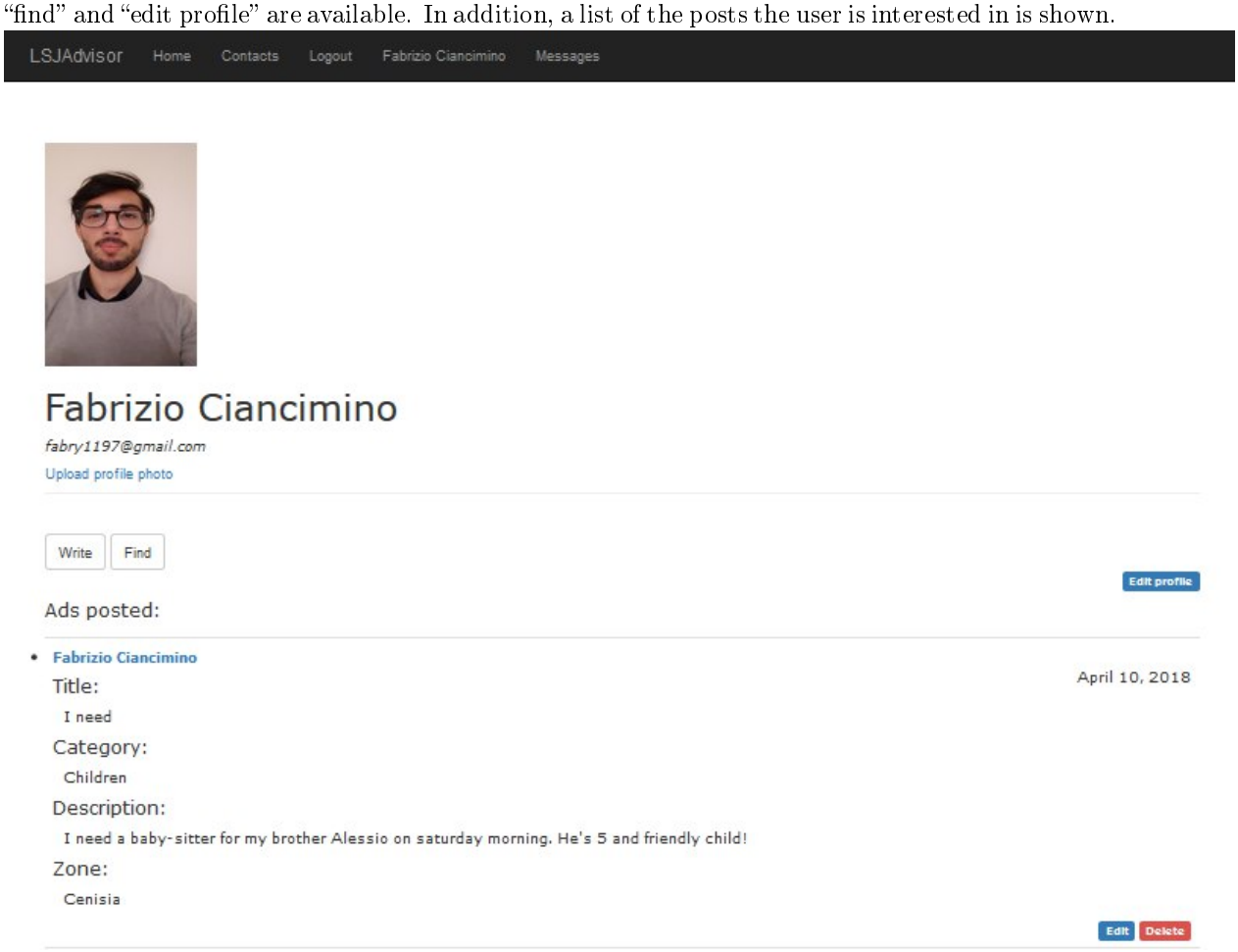
Nowadays hard work and study make even the most simple routine activities difficult, and this make time management more complicated. LSJ Advisor aims to help students, workers and whoever has to face this problem. People who desire to entrust these tasks to someone will be able to find help in an easier and faster way. At first you should see if there is someone who is looking for activities to earn some money and who is available to help you. If you can't find anyone who could match your issue, create a new post that informs people about what kind of activity they can perform in your place. The more details the better, as it helps people know if they are available to help you. Do not waste your time, use LSJ Advisor.

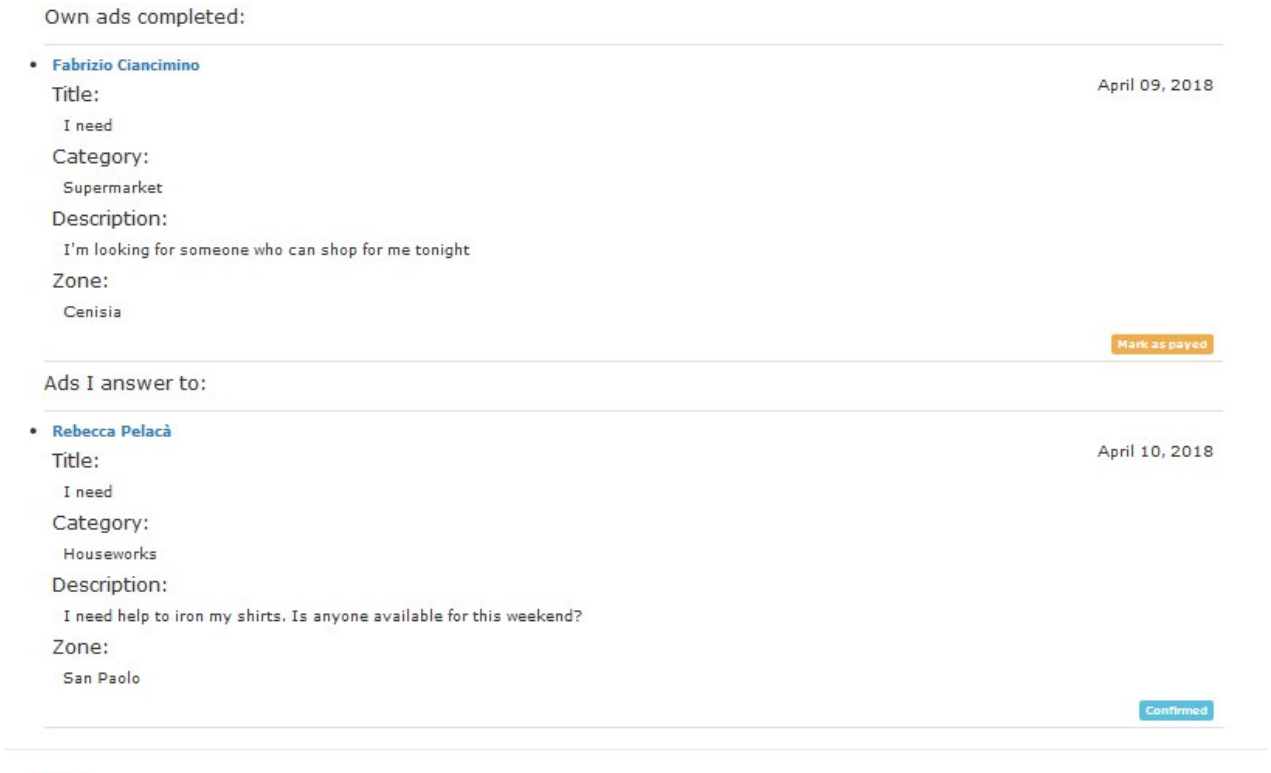
1.2 LSJ Insight

Before explaining the creation and development of the website, we will give a short introduction into the site itself. The following graphic shows the sign up process and the general layout of LSJ Advisor. Any users can create an account entering their personal information (name, surname and personal e-mail).



After registering or log in, the user will be redirected to its profile page in which the "write", "find" and "edit profile" are available. In addition, a list of the posts the user is interested in is shown.





From these pages the user can add or find a post if he is looking for help, or try to find someone who requires his services.

2 Feasibility Study

The following paragraphs show the current state and the results processed, according to the study we carried out through the use of the main qualitative methods, as well as the development process we have gone through during the project.

2.1 Stakeholder Analysis

The stakeholders who might affect or be affected by our system were identified and sorted through a careful study of their possible interactions, i.e. the study of their impact on the action and/or the impact the action will have on them. The analysis aims at fixing our risk management and was a helpful tool for strategic decision making.

2.1.1 Stakeholder Matrix

The stakeholder matrix is a very effective tool for analysing stakeholders. Stakeholders include individuals, community leaders, groups and other organisations who will be impacted by the program, or who could influence the outcome. They can be internal or external. The external stakeholders are divided into two categories namely Final users and Business partners, whereas the IT staff and Management represent the internal stakeholders. Management consists of us, and all team members who are responsible for strategic decision-making and business development. IT staff consists of technicians and support staff who are concerned with software development and web design. The advertising partners that belongs to the category of 'Business partners' are being considered as potential stakeholders in the future. In this way, our website can provide a greater visibility to them.

Stakeholders Group	Interests	Capacity and motivation to bring about change	Possible actions to address stakeholder interests
External Stakeholders			
Final users			
Students and workers who are looking for help: Students and workers who do not have enough time or skills	• Saving time • Saving money • Meeting new people	• Investing free time in more interesting activities • Will to meet new friends • Learning new skills	• Cheap service • An easy and fast way to connect people • An alternative method for organizing routine works
Students, part-time workers or unemployed who offer their availability	• Earning money • Meeting new people • Finding new jobs	• Increasing worker's wages • Meeting new friends • Showing own skills	• Occasional work • Low-skilled job
Business partners			
Government Institution that provides financial law and legislation	• Taxation • Legalization & compliance with Italian law	• Providing rules to safeguard supply and demand	• Tax payment
Post Office	• Decreasing the flow of people	• Increase of service efficiency	• Decreasing the queue
Schools	• Decrease of personnel's cost • Less responsibility for controlling pupils • Reduction of the number of school bus	• Providing alternatives to increase the revenues • Decreasing of the cost for school bus service	• Providing a safe lift for children
Advertising Partners	• Growth their visibility • Increasing their profit	• Monetary support	• Promoting their offering
Internal stakeholders			
Management Accounting & Finance Marketing Business Development	• Managing the business to achieve the strategic objectives • Increasing profits	• Implementing the best managerial performance • Reward system	• Managing Financial Accounts • Achieving the target revenue • Marketing our Brand
IT staff Web designing Software development	• Increasing website traffic • Increasing the SEO (Search Engine Optimization)	• Reward system • Unleash their potential competences • Career development	• User-friendly browsing interface • Flexibility • Accessibility

2.1.2 SWOT Analysis

SWOT Analysis (Strengths, Weakness, Opportunities, Threats) is an organized list. It's a useful technique for understanding your Strengths and Weaknesses, and for identifying both the Opportunities open to you and the Threats you face. Strengths and weaknesses are characteristics of the project: they outline advantages or drawbacks disadvantages over our competitors. Opportunities and threats are elements in the environment that can either be exploited to our benefit or can have a negative effect on the project. We have considered the framework as a powerful support for our decision-making as it enables an entity to uncover opportunities for success that were previously unarticulated or to highlight threats before they become overly burdensome.

Streghts:

- Creation of a unique market place for occasional low skilled works
- Free service for the users
- Management of LSJ advisor is composed by potential users with this perception of customers needs
- Develops are rating network for new and experience users
- Offering diversified services
- Dematerializing connections between users

Weakness:

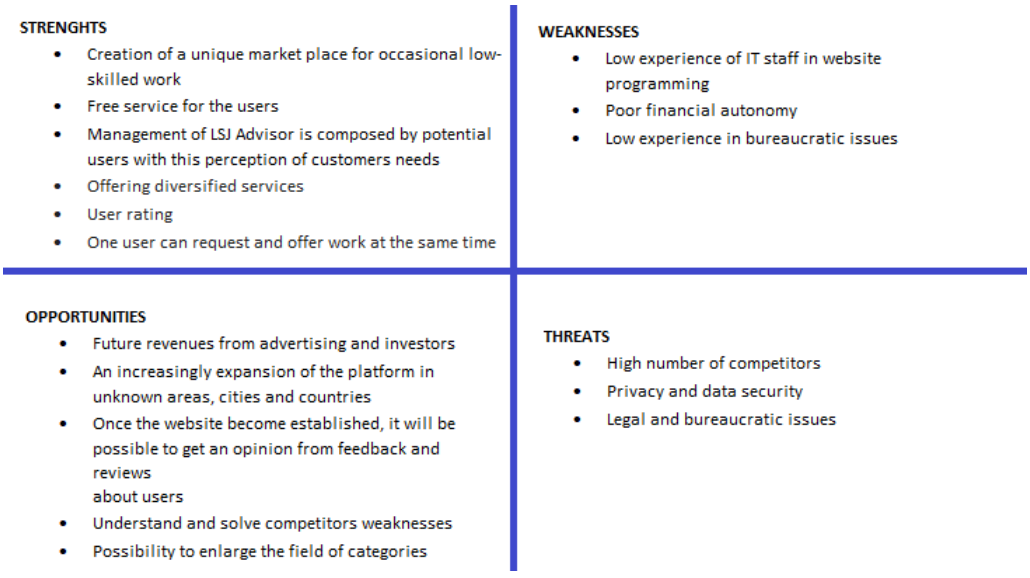
- Low experience of IT staff in website programming
- Poor financial autonomy
- Low experience of bureaucratic issues

Opportunities:

- Future revenues from advertising and investors
- An increasingly expansion of the platform in unknown areas, cities and countries
- Once the website become established, it can be possible to get an opinion from feedback and reviews about trips and users
- Understand and solve competitors weaknesses

Threats:

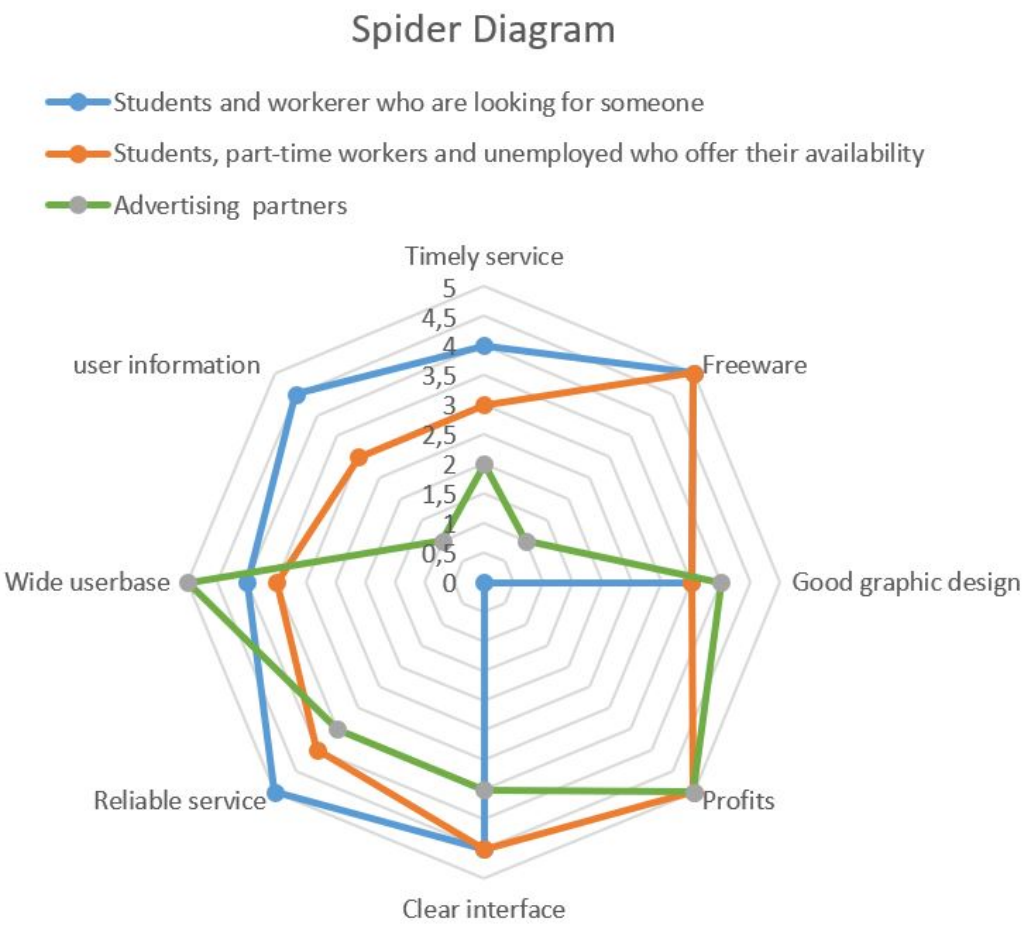
- High number of competitors
- Privacy and data security
- Legal and bureaucratic issues



2.1.3 Spider Diagram

Spider Diagram is used to analyse and identify the main aspects that managers have to focus on in order to provide an efficient service. We have selected three stakeholders (advertising partners and both students and workers who need or offer a service) and eight requirements. Then, a score from 0 to 5 has been assigned based on the relevance of these for the stakeholders. Advertising partners were included because we consider advertisement as a potential source of income in the future. The requirements considered are the following:

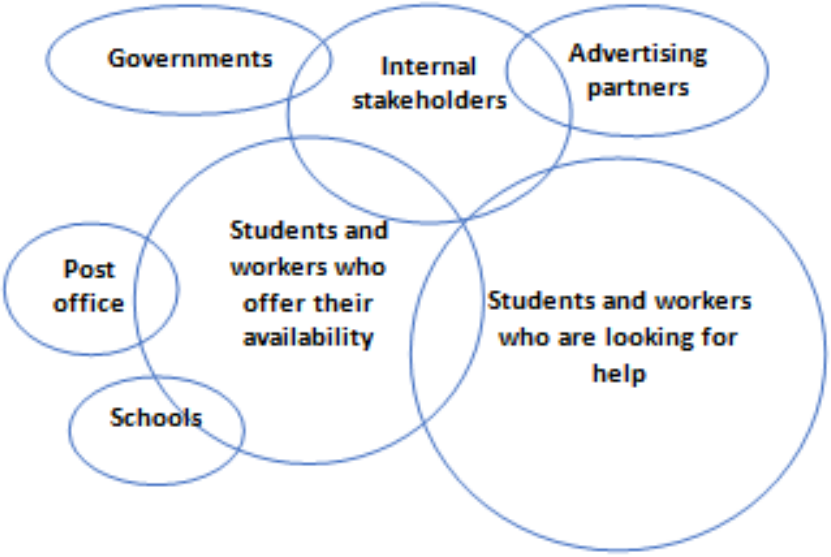
- Timely service
- Freeware
- Good graphic design
- Profits
- Clear Interface
- Reliable service
- Wide user base
- User information



Final users both prioritize the reliability of the service and to use a freeware. Moreover, students who offer their availability and advertising partners are very interested on profits. The advertising partners value the size of user base as a key attribute.

2.1.4 Venn Diagram

The Venn diagram highlights the relationship between all the stakeholders found out in the Stakeholder matrix. The size of the circles is due to the importance of each stakeholder, whereas the overlap indicates the degree of the interaction.

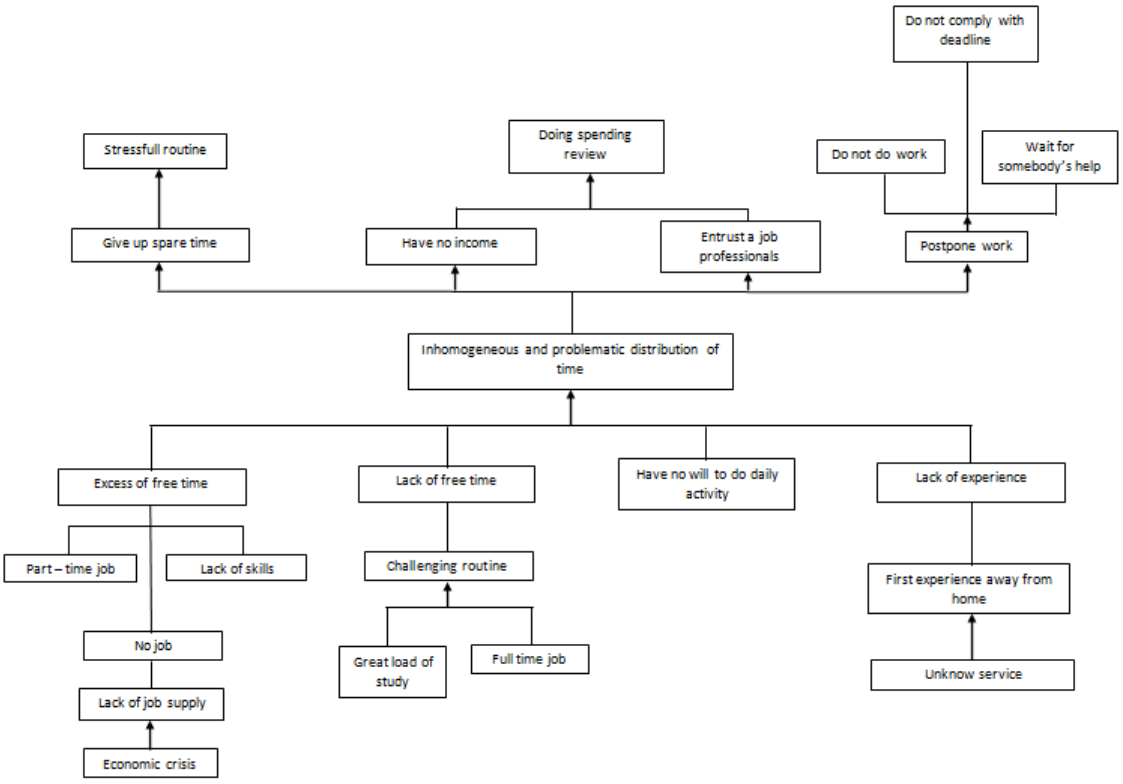


2.2 Problem Analysis

Problem posing is the first step in the successful execution of a project. Firstly, a problem analysis is done in order to identify all the negative aspects of the current situation regarding the stakeholders. Moreover, the analysis is used to develop a cause and effect relationship between the identified problems. Problem Tree is used to represent the hierarchy of problems. Analysing the problems that a potential user of LSJ Advisor could face, we laid out the following tree diagram.

2.2.1 Problem Tree

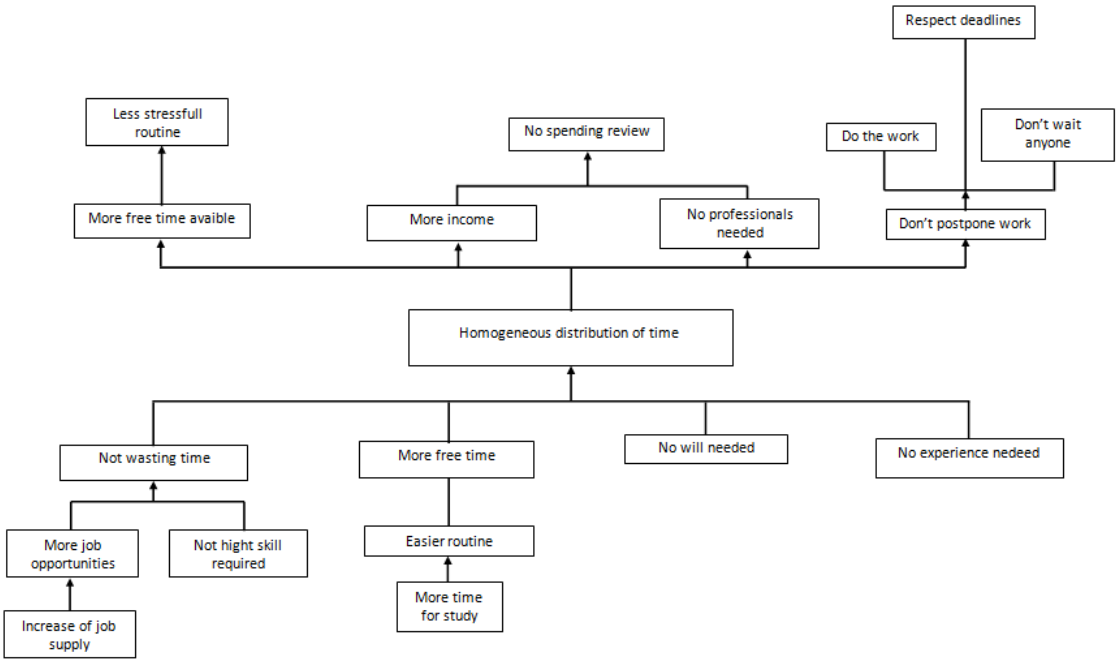
The following tree provides the hierarchy of the problems that a potential customer of LSJ Advisor could face, according to the results of the analysis made.



We found that the main problem is the inhomogeneous and problematic distribution of time. This main issue has been posed in the centre of our diagram. Since LSJ Advisor has been designed in order to provide a solution to this problem, it represents a marketplace for this kind of services. As a result of this work, people have the possibility to purchase or sell occasional services. Joining the platform, people that request and those who offer service can easily get in contact.

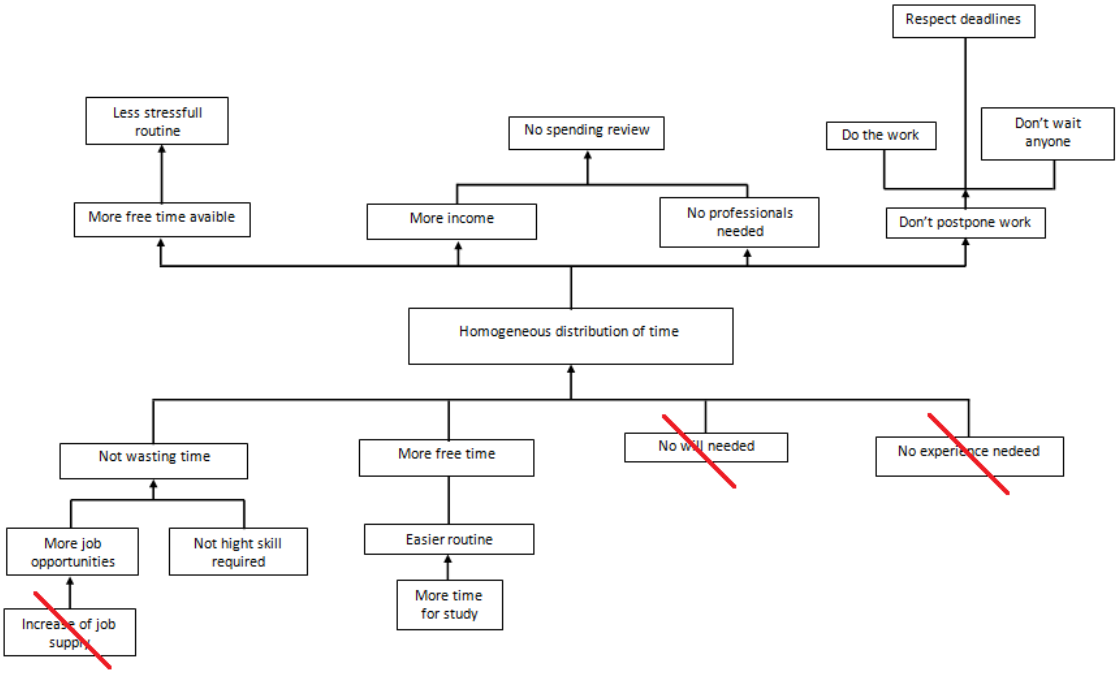
2.3 Solution Analysis

Starting from the problem tree, we created a new tree that contains solutions to the problems spotted in the previous diagram. This new tree is organized in three levels referring to the "overall objectives" of our work, the main "purpose" and the results expected. All of these are organically linked through relation edges. The following graphic representation displays the outcome.



2.4 Strategy Selection

As we pointed out the results in the solution tree, the next step consists in choosing which results are really influenced by LSJ Advisor, cutting all those solutions which are not affected by our project.

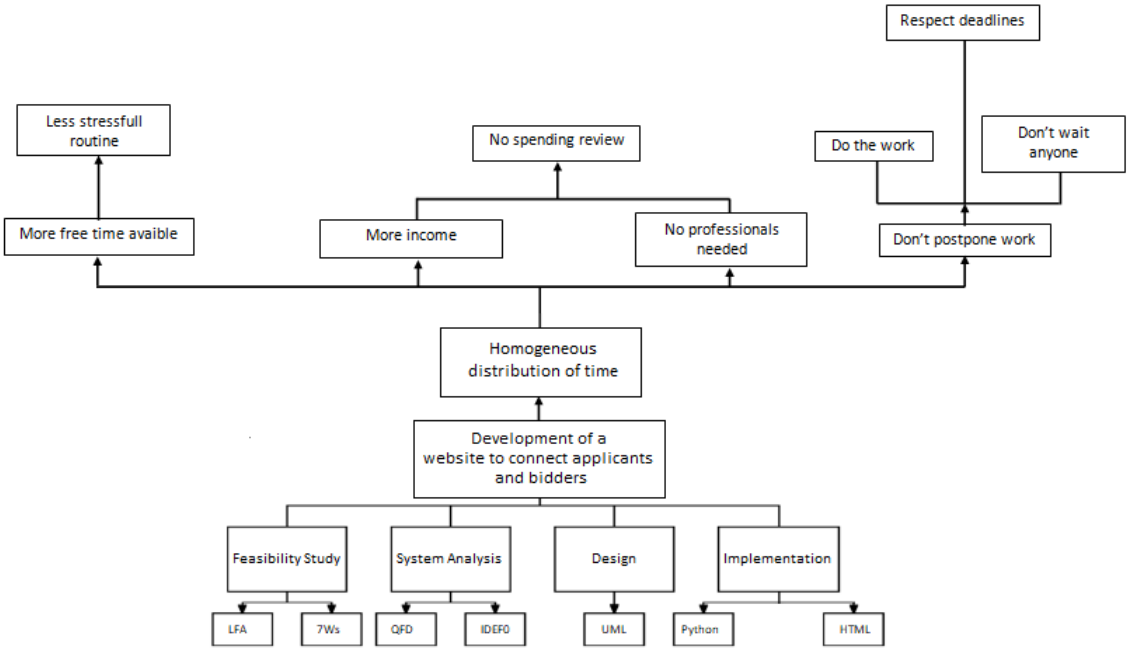


As already mentioned, the last tree underlines the expected effects of LSJ Advisor and outlines the strategy we want to carry on. If students or workers desire to demand or offer occasional low-skilled job, the website provides the opportunity to get in communication the two users.

2.5 Objectives Analysis

Finally, we selected the most suitable tools for our website implementation. At the current point, we need to do:

- Feasibility Study, to be completed through the development of the LFM and the activity scheduling;
- System Analysis, developing requirements coming from customers' needs and pictures of all processes our platform has to include;
- Design the steps, that gives a visual design of the system;
- Implementation, including programming languages on which the website will be built.



2.6 Logical Framework Matrix

The Logical Framework Matrix assists in defining the pathway by which objectives will be reached, identifying the potential risks in achieving them, and provides also a summary of the activities. A Logical Framework Matrix was created in order to have an overview on how various aspects of the project are logically linked to each other. In the first columns it is found the hierarchy of objectives all the way down to the activities that need to be carried out. Starting from the bottom of the first column, the corresponding assumptions in the same row are considered to be true in order to reach the objective. The second and third column represent the Indicators and the Sources of Verifications that are important for monitoring and evaluation throughout the project. Performance and Perception indicators were used in order to execute periodic controls of the platform.

Performance	Perception
• N° of users • N° of answered ads • N° of shared ads • N° of site visits • N° of bugs in the website	• Users feedback • Advertising partner

Project description	Indicators	Sources of verifications	Assumption
Overall objective: To provide an easy interconnection between people who offers and/or requests an occasional work in the neighbourhood	• People obtain more free time • People gain more money	Analysis through periodic surveys and questionnaires	
Purpose: Users obtain more free time or the possibility to gain more money	• Users feedback • N° of answered ads / N° of shared ads • N° of visits of the website	• User feedback • Counter for website visits • Counter for answered ads • Counter for shared ads	Users use the website and it has a positive impact on their free time and routine
Results: Create a website for an interconnection between offerers and applicants about an ordinary works	• N° of registrations • N° of bugs in the website		• Users friendly interface with accessible function • Recking the most important costumer requirement
Activities: LFA 7Ws QFD IDEF0 UML Implementation			• User friendly interface with accessible functions • A large number of users seeking for company

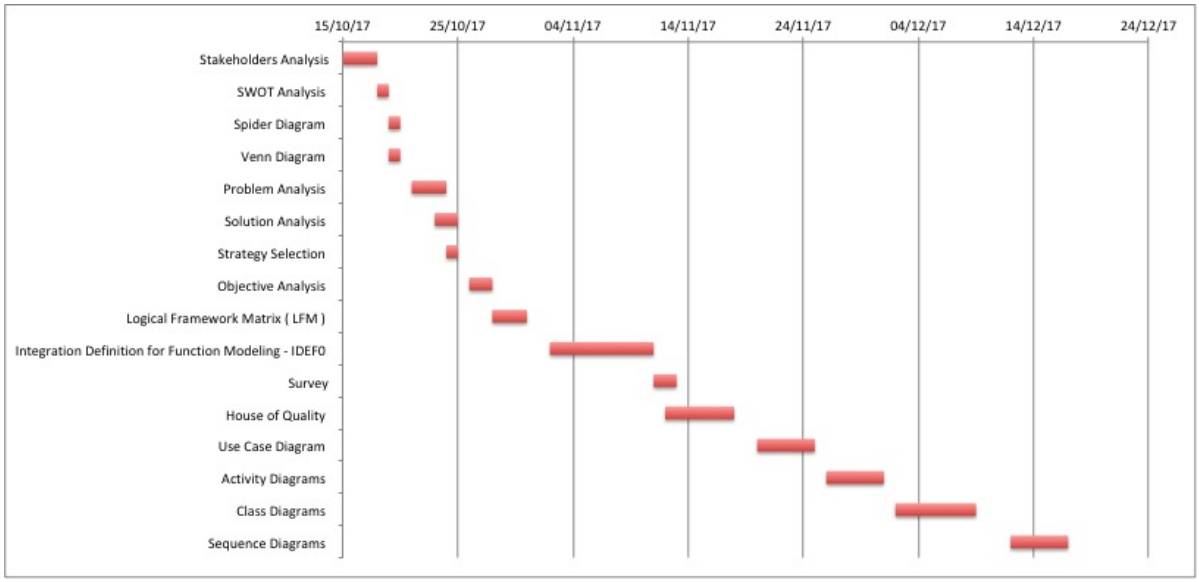
2.7 Project Scheduling

Project scheduling is a mechanism to communicate what tasks and the timeframe in which each part of project has to be completed. The order is not only chronological but also logical since proceeding with one activity requires the accomplishment of the one before.

	start	end	days
Project Scheduling	15/10/17	16/12/17	63
Logical Framework Analysis - LFA	15/10/17	30/10/17	16
Stakeholders Analysis	15/10/17	17/10/17	3
SWOT Analysis	18/10/17	18/10/17	1
Spider Diagram	19/10/17	19/10/17	1
Venn Diagram	19/10/17	19/10/17	1
Problem Analysis	21/10/17	23/10/17	3
Solution Analysis	23/10/17	24/10/17	2
Strategy Selection	24/10/17	24/10/17	1
Objective Analysis	26/10/17	27/10/17	2
Logical Framework Matrix (LFM)	28/10/17	30/10/17	3
Integration Definition for Function Modeling - IDEF0	02/11/17	10/11/17	9
Quality Function Deployment - QFD	11/11/17	17/11/17	7
Survey	11/11/17	12/11/17	2
House of Quality	12/11/17	17/11/17	6
Unified Modeling Language - UML	20/11/17	16/12/17	27
Use Case Diagram	20/11/17	24/11/17	5
Activity Diagrams	26/11/17	30/11/17	5
Class Diagrams	02/12/17	08/12/17	7
Sequence Diagrams	12/12/17	16/12/17	5

2.7.1 Gantt Chart

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph show the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project.



3 Life Cycle Development

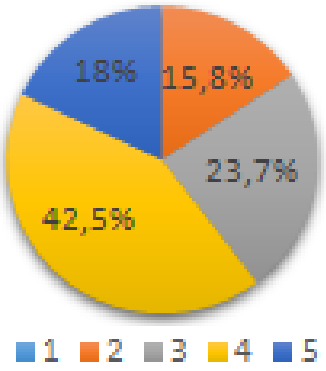
This part of the analysis handles the more theoretical reasoning made before and presents a rather operational and practical examination of the project, with the help of various tools.

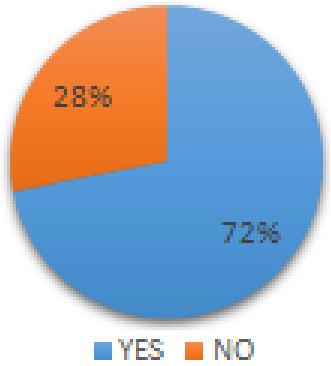
3.1 7Ws Analysis

Why?	*"I like work: it fascinates me. I can sit and look at it for hours"* – Jerome K. Jerome
What?	A platform to gather people who offer and/or request an occasional low skilled service
Who?	Students and workers. Also, business partners can be involved.
Where?	The Internet
When?	By 2018
How?	Brainstorming sessions, looking at student's issues, focus groups, planning section, learning and implementation of the code.
How much?	Time, effort and pursuance for the team.

3.2 Quality Function Deployment

Quality Function Deployment (QFD) is a structured approach to define customer needs or requirements and translating them into specific plans to produce products that meet those needs. The "voice of the customer" is the term to describe these stated and unstated customer needs or requirements. The QFD method is a powerful tool in the product development and quality engineering branch, indeed it represents a big part of the evolution we have gone through during the project. The following QFD report puts a strong emphasis on the input values, represented by customer requirements, technical characteristics and competitor benchmarks. To have a representative overview about the requirements we have interviewed potential future customers with different backgrounds and needs. The feedback from the interviews led us to 10 requirements that were critical for our potential clients. As a second step, the team members rated these requirements by giving them points in 1-5 range depending on their importance. The results are shown in the left-hand side pie chart. According to the most significant results of the survey, this pie chart displays how our customers strongly consider useful feedbacks and users reviews from other users. Another striking thing is that 72% of our costumers are willing to offer or request low skilled service to unknown people, showing us the potential growth of the website thanks to the solid base idea of it. Finally, given the surveys' results, we decided to consider the most popular needs to construct our QFD.





3.2.1 Customer Requirements (Whats)

In order to understand requirements that customers would have about the new product, we conducted a interview. Then we collected raw data to highlight the so called Voice of the Customer (VoC). This helped us to identify all product features that could be included in our project and that would give more satisfactions to users. The most important needs come up from the survey are:

- Free service
- Security & Privacy
- Large user base
- Quick response to ads
- Simple interface
- Lots of ads
- Users reliability
- Rating assessments for a better choice
- Degree of advertisement
- Good performance of the website

3.2.2 Technical Features (Hows)

After determining "what" users desire, we focused on "how" we could reach those objectives. Some measurable and controllable design requirements/characteristics/parameters, influencing customer satisfaction when modified, have been identified and they describe the Voice of Engineer (VoE). They are:

- Database
- Server speed
- Server capacity
- HTML/MySQL/Python
- Revenue generating model
- Creation accounts
- Email verification
- Ads deadline

3.2.3 Relationship Matrix

The relationships between "Hows" and "Whats" are depicted by the "Relationship Matrix" where the importance of the correlation is given by specific symbols. A strong relation implies that even a small variation in the engineering characteristic results in an important variation of customer satisfaction of his need. For instance, if the server speed was not very performing, users would be unsatisfied and shift to our competitors.

	Degree of importance	Relative Importance	Database	Server speed	Server capacity	HTML/MySQL/Python	Revenue generating model	Creation accounts	Email verification	Ads deadline
Free Service	4,6	12,30%		▲	▲		●	▲		
Security & Privacy	4,1	10,96%	◎					●	◎	
Large User Base	4	10,70%	●		●		◎			
Quick Response	3,9	10,43%		▲	▲			▲		●
Simple Interface	3,5	9,36%				●				
Lots of Ads	3,4	9,10%	▲		▲		▲			◎
User Reliability	4,4	11,76%	▲					◎		
Rating assessments for a better choice	3,9	10,43%				●				
Degree of Advertising	1,6	4,30%		◎	◎		●			
Good Performance of the Website	4	10,70%	●	●	◎	●				▲
	37,4	100%								

3.2.4 Planning Matrix

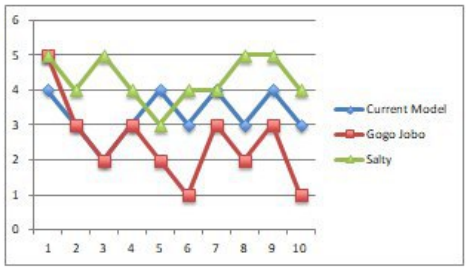
After defining users' requests, customers' perceived quality of our current product and our competitors is evaluated and ranked, using a scale from 1 to 5. This is the so called Benchmarking, useful to understand where we have to invest more in order to increase their satisfaction.

Requirement Satisfaction
1 = Very unsatisfied
2 = Unsatisfied
3 = Relative satisfied
4 = Satisfied
5 = Very satisfied
Customer Importance
1 = Not important
2 = Minor importance
3 = Some importance
4 = Strong importance
5 = Very strong importance

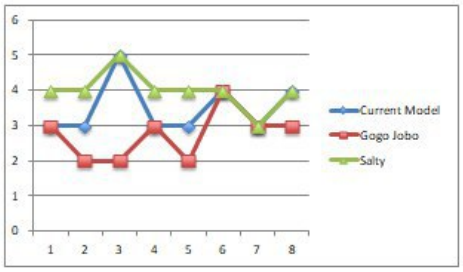
	Current Model	Gogo Jobo	Salty	New model	Improvement ratio	Strenght	Absolute weight	Relative weight
Free service	4	5	5	5	1,25	1,5	7,5	14,65%
Security & Privacy	3	3	4	4	1,33	1,2	4,8	9,38%
User reliability	4	3	4	5	1,25	1,2	4,8	9,38%
Good performance	3	1	4	4	1,33	1,2	4,8	9,38%
Simple interface	4	2	3	4	1	1	4	7,81%
Find a lots of ads	3	1	4	4	1,33	1	4	7,81%
Rating assessments	3	2	5	5	1,66	1	5	9,77%
Degree of advertising	4	3	5	4	1	1,5	7,5	14,65%
Quick response	3	3	4	4	1,33	1,2	4,8	9,38%
Large user base	2	2	5	4	1,2	1	4	7,81%

The absolute weight of each need is computing by summing the product of degree of importance, how much influencing that need is from our point of view, the improvement ratio, ratio between our target and the current model, ranking of customer opinions, and strength. Following there are graphs comparing the Benchmarking of the perceived and offered quality respectively on the left and on the right. We ranked those values from competitors' analysis and from asking people to create an account on the other platforms and to test them as they are customers.

	Current Model	Gogo Jobo	Salty
Free Service	4	5	5
Security & Privacy	3	3	4
Large User Base	2	2	5
Quick Response	3	3	4
Simple Interface	4	2	3
Lots of Ads	3	1	4
User Reliability	4	3	4
Rating Assessments	3	2	5
Degree of Advertising	4	3	5
Good Performance of the Website	3	1	4

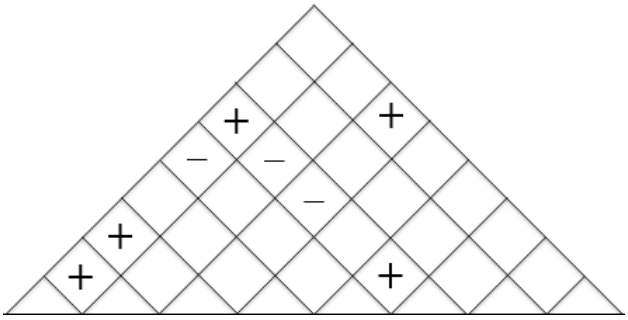


	Current Model	Gogo Jobo	Salty
Database	3	3	4
Server Speed	3	2	4
Server Capacity	5	2	5
HTML/MySQL/Python	3	3	4
Revenue Generating Model	3	2	4
Creation Accounts	4	4	4
Email Verification	3	3	3
Ads Deadline	4	3	4



3.2.5 Correlation Matrix

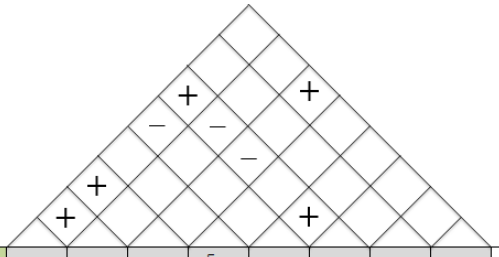
The Correlation Matrix is located on the top of House of Quality and allows to describe the relationship between technical features by using qualitative symbols representing the negative or positive trend of each correlation. There are some features which support each other and others that get in each other's way. For instance, if the number of registrations is increased, the server capacity decreases consequentially.



Key to Roof/ Correlation Matrix Symbol	
+	Positive / Supporting
-	Negative / Trade off

3.2.6 House of Quality

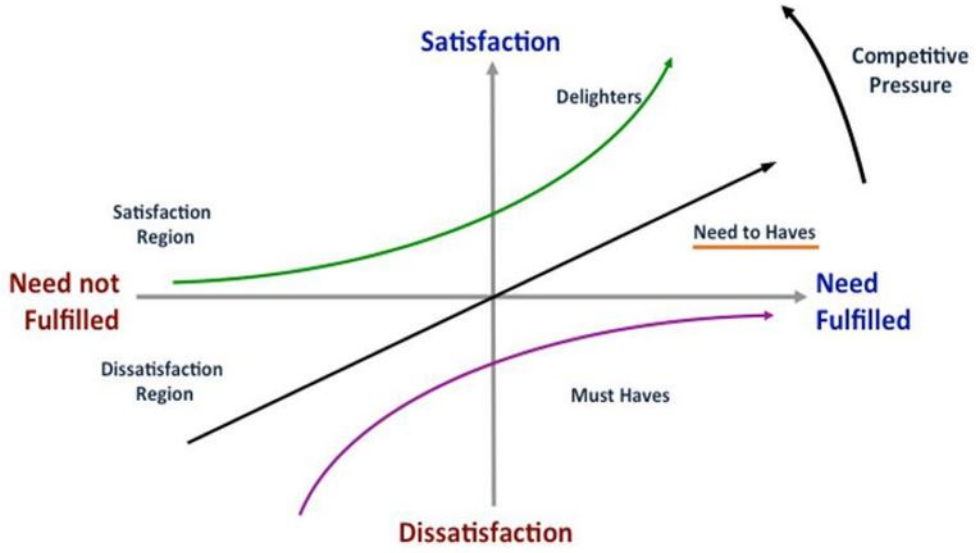
Here the whole House of Quality is reported. It is the graphic result of the QFD analysis with the strengths and weakness of our project and for understanding better what customers want and need.



	Degree of importance	Relative Importance	Database	Server speed	Server capacity	HTML/MySQL/Python	Revenue generating model	Creation accounts	Email verification	Ads deadline	Current Model	Gogo Jobo	Salty	Target for the New Model	Improvement Ratio	Strenght	Absolute Weight	Relative Weight
Free Service	4,6	12,30%		▲	▲		●	▲			4	5	5	5	1,25	1,5	7,5	14,65%
Security & Privacy	4,1	10,96%	◎					●	◎		3	3	4	4	1,33	1,2	4,8	9,38%
Large User Base	4	10,70%	●		●		◎				2	2	5	4	2,00	1,2	4,8	9,38%
Quick Response	3,9	10,43%		▲	▲			▲		●	3	3	4	4	1,33	1,2	4,8	9,38%
Simple Interface	3,5	9,36%				●					4	2	3	4	1,00	1	4	7,81%
Lots of Ads	3,4	9,10%	▲		▲		▲			◎	3	1	4	4	1,33	1	4	7,81%
User Reliability	4,4	11,76%	▲					◎			4	3	4	5	1,25	1	5	9,77%
Rating assessments for a better choice	3,9	10,43%				●					3	2	5	5	1,67	1,5	7,5	14,65%
Degree of Advertising	1,6	4,30%		◎	◎		●				4	3	5	4	1,00	1,2	4,8	9,38%
Good Performance of the Website	4	10,70%	●	●	◎	●				▲	3	1	4	4	1,33	1	4	7,81%
	37,4	100%															51,2	100,00%

Technical importance	92,1	49,3	64,7	102,6	71,2	58,6	12,3	49,3	500,1
Relative importance	18,42%	9,86%	12,94%	20,52%	14,24%	11,72%	2,46%	9,86%	100%
Absolute weight	2,00	1,22	1,68	2,72	2,52	1,38	0,28	1,16	12,97
Relative weight	15,45%	9,44%	12,94%	21,01%	19,45%	10,62%	2,17%	8,92%	100%

Going through the analysis of customer needs we found out that to reach the maximum satisfaction of customers, we do not need to satisfy only basic needs, that have to be met in any case without user asks, such as a minimum degree of users' privacy. Assumed that if we want to increase the satisfaction of users, we need to fulfill higher needs come from expectations that have been already set and that must be met by enhancing the performances of the website. Beyond basic and performance needs, there is a kind of needs that customers sometimes do not expected to have and that they will make their satisfaction higher than before. The "delighters" are our aim and in general of each company in the world. The one showed in the picture is a simple diagram built up by Noriaki Kano and intended for understanding business customer needs.



3.2.7 Competitor Analysis & Benchmarking

Competitors that offer the same service:

- Gogojob: www.gogojobo.it
- Salty: www.saltyapp.com

Competitors that offer different service but may satify the same needs:

- Subito: www.subito.it
- Indeed: it.indeed.com

Competitors Understanding strengths & weaknesses:

- www.gogojobo.it
  - Strenghts :
    - Link Facebook account
    - Free Service
    - Link "jobber" and "boss"
    - Users can be "jobber" and "boss" at the same time
    - Clear interface
    - Unlimited kinds of jobs
  - Weaknesses :
    - Doesn't work
    - Available only for Android
- www.salty.com
  - Strenghts :
    - Free service
    - Simple and innovative interface
    - Link Facebook account
    - Communication through chat messages

    - Communication through private phone
    - Feedback and rating valuation
    - Many parameters of research
  - Weaknesses :
    - Only for professionals
    - No professional certification is required
    - Few service available
    - Bug in creation account
    - Bug in updating profile

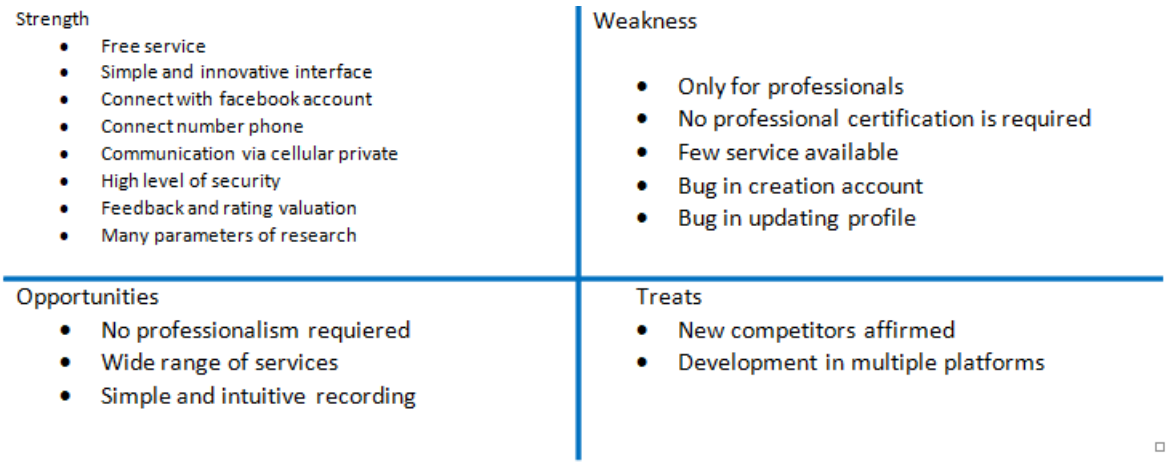
SEO Analysis :

- We used the following SEO Analyser tools.neilpatel.com:

	www.salty.com	www.gogojobo.it
ESTIMATED TRAFFIC SCORE	78	59
SPREED SCORE	75	64
SEO SCORE	72	82
Total Number of Warnings	4	4
Load Time	5.51 sec	4.43 sec

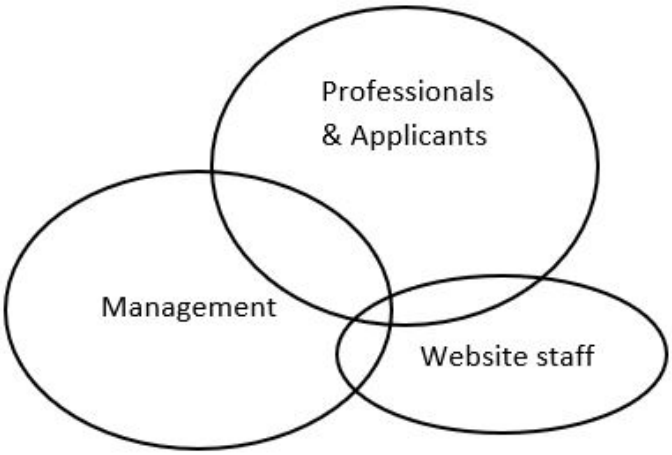
3.2.8 Salty: SWOT Analysis

A SWOT analysis was performed in order to clearly lay down strengths, weaknesses, opportunities and threats of our main competitor: Salty. Strengths and weaknesses are characteristics of the competitor that we are analyzing through an "Internal" analysis: they outline advantages or a disadvantages over its competitors. Opportunities and threats are elements in the "External" analysis related to Salty.

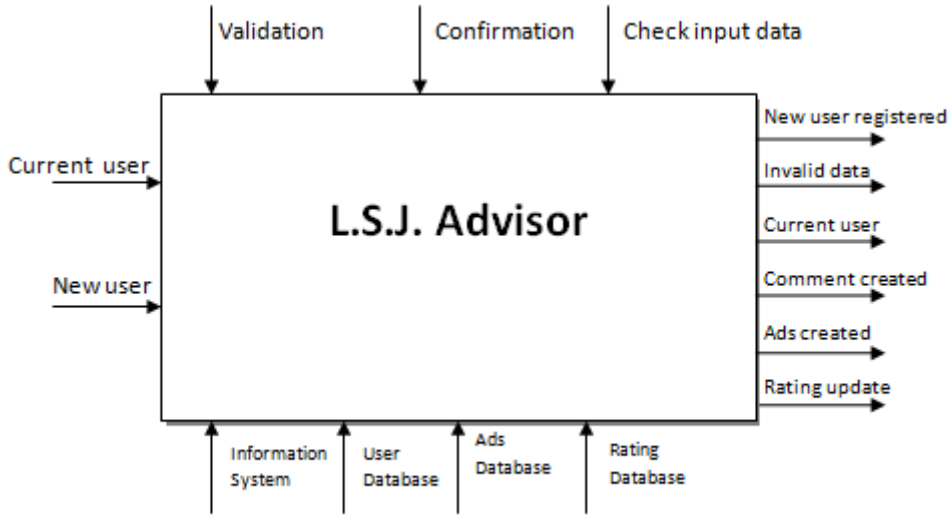


3.2.9 Salty : Venn Diagram

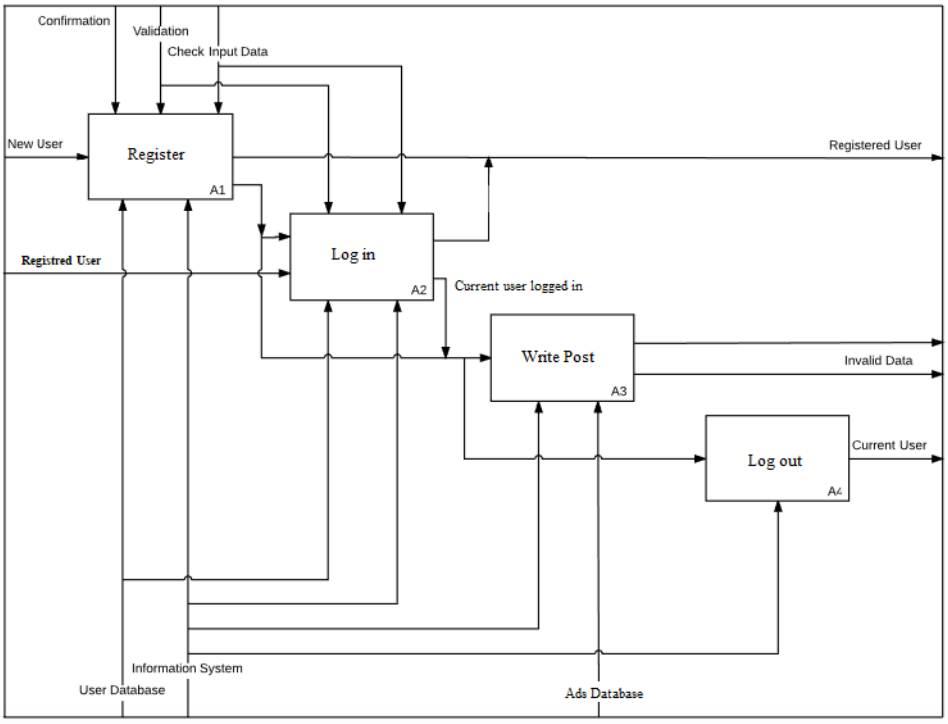
A Venn diagram was created to highlight the nature of relationships that exist between all the stakeholders that the Competitor Salty is identified in his description matrix. The dimensions of the circles illustrate the relative importance of the stakeholders and the overlap signifies their interaction.



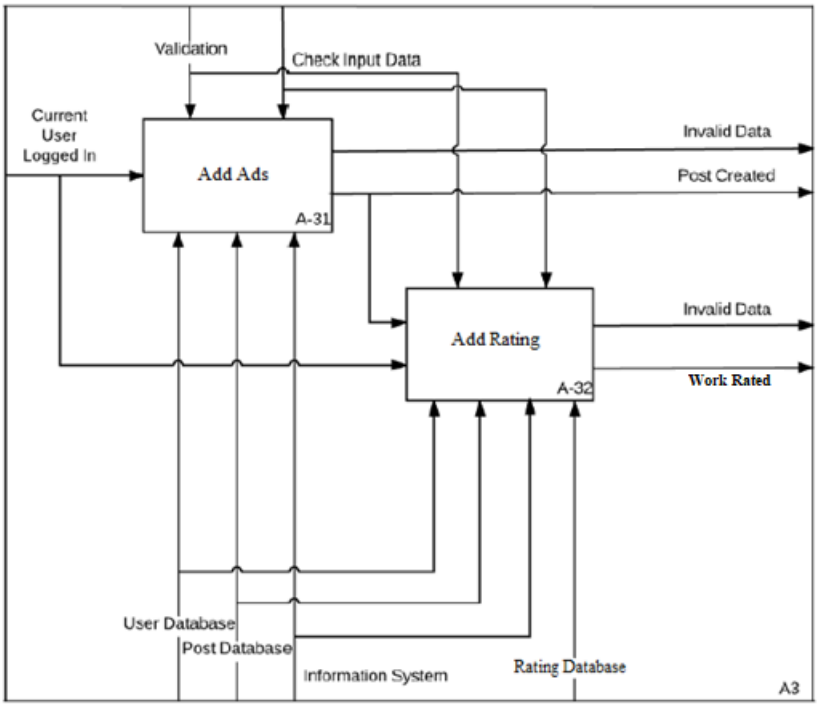
3.3 TO-BE IDEF0 Analysis



An IDEF0 analysis was made in order to visualize the various functions of the platform and what is needed to perform those functions. It defines the inputs, outputs, resources and limitations of each process. Therefore, it helped us to understand the service we are going to provide before its implementation. Every IDEF0 box receives some inputs from the left side, which are items that activate the process within the box. On the top we have the incoming controls that are used to direct or regulate the activity in the process. On the bottom side we have the mechanisms, which enable and perform an activity using people, system, data and equipment. Finally, on the right side, there are the outputs, represented by outgoing arrows, that describe what is produced by an activity. The platform is thought to be minimalistic, in order to reflect the simplicity of the provided service. This can be observed in the following analysis as the service is divided into four main functions along with some minor basic functions.



New users will have to access the Register function and sign up using their personal email. During the registration the website, to be sure that the user doesn't insert wrong data, controls that the email is written in the right way and that the username chosen doesn't exist yet in the database. Existing users will have an option to sign in and directly access the service. The major function of the platform is called Write Post. It can be accessed by registered users. The user is able to write an ad for his/her need. After the low skilled job has been done, user can write a comment and rate the person who has completed the work.

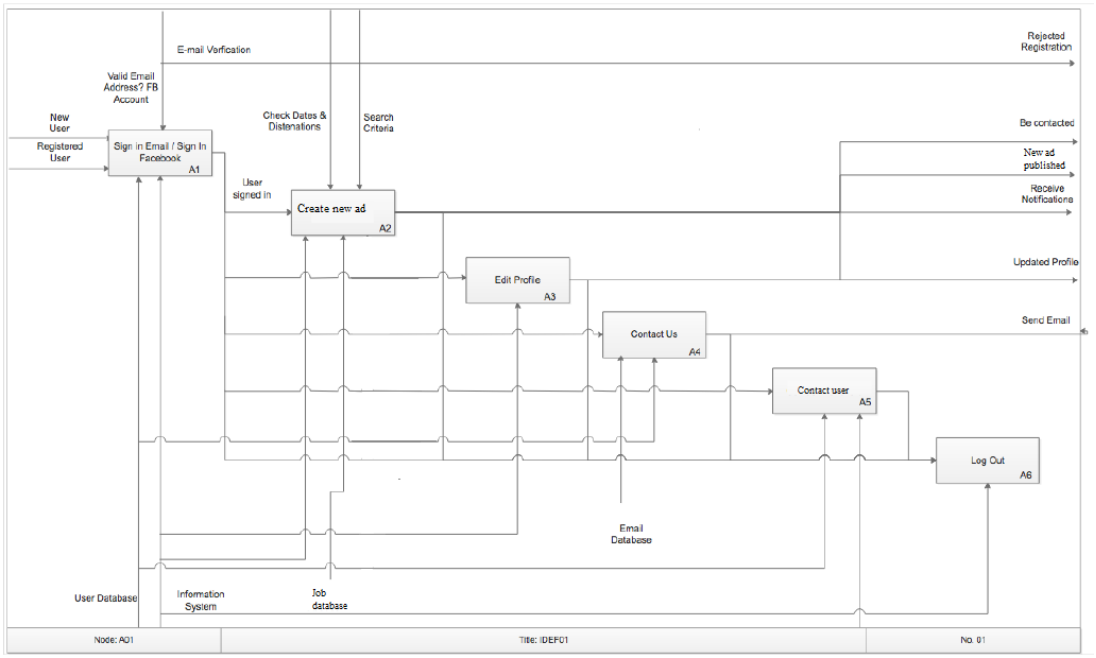


The function A3 'Write Post' is further divided into two functions 'Add Ads' and 'Add Rating'. Since our web application is not unique in its kind, but we have competitors operating in the same

field, not only a To-Be IDEF0 analysis was performed but also an AS-IS IDEF0 analysis for our main direct competitor: Salty.

3.4 AS-IF IDEF0 Analysis

An IDEF0 analysis was made in order to better visualize the various functions of the platform and clearly define the inputs/ outputs/ resources and limitations of each process. The platform is thought to be minimalistic in order to reflect the simplicity of the service – which is to connect people who respectively demand and offer time to carry out a low-skill job. This can be observed in the following analysis as the service is divided into four main functions "Create new ad", "Edit Profile", "Contact Users", "Contact Us", along with minor basic functions.

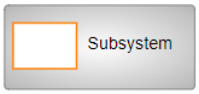


4 UML Diagrams

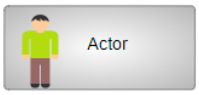
The Unified Modelling Language (UML) is a general-purpose, developmental, modelling language in the field of software engineering, which is intended to provide a standard way to visualize the design of a system. Proceeding with the design phase of the project, UML was used in order to graphically represent different aspects of the website in a standardized way, including elements such as activities, individual components of the system and how they can interact with other software components, how entities interact with others (components and interfaces) and external user interface. After completing the UML structure, we used the diagrams that were created as the basis of the coding activity.

4.1 Use Case Diagram

The first step was the completion of the UML Use Case Diagram. It provides a basic structure, showing in which way the different users (or actors) who interact with it can access the system of L.S.J Advisor. What follows is the explanation of the UML standard notification applied to this project.

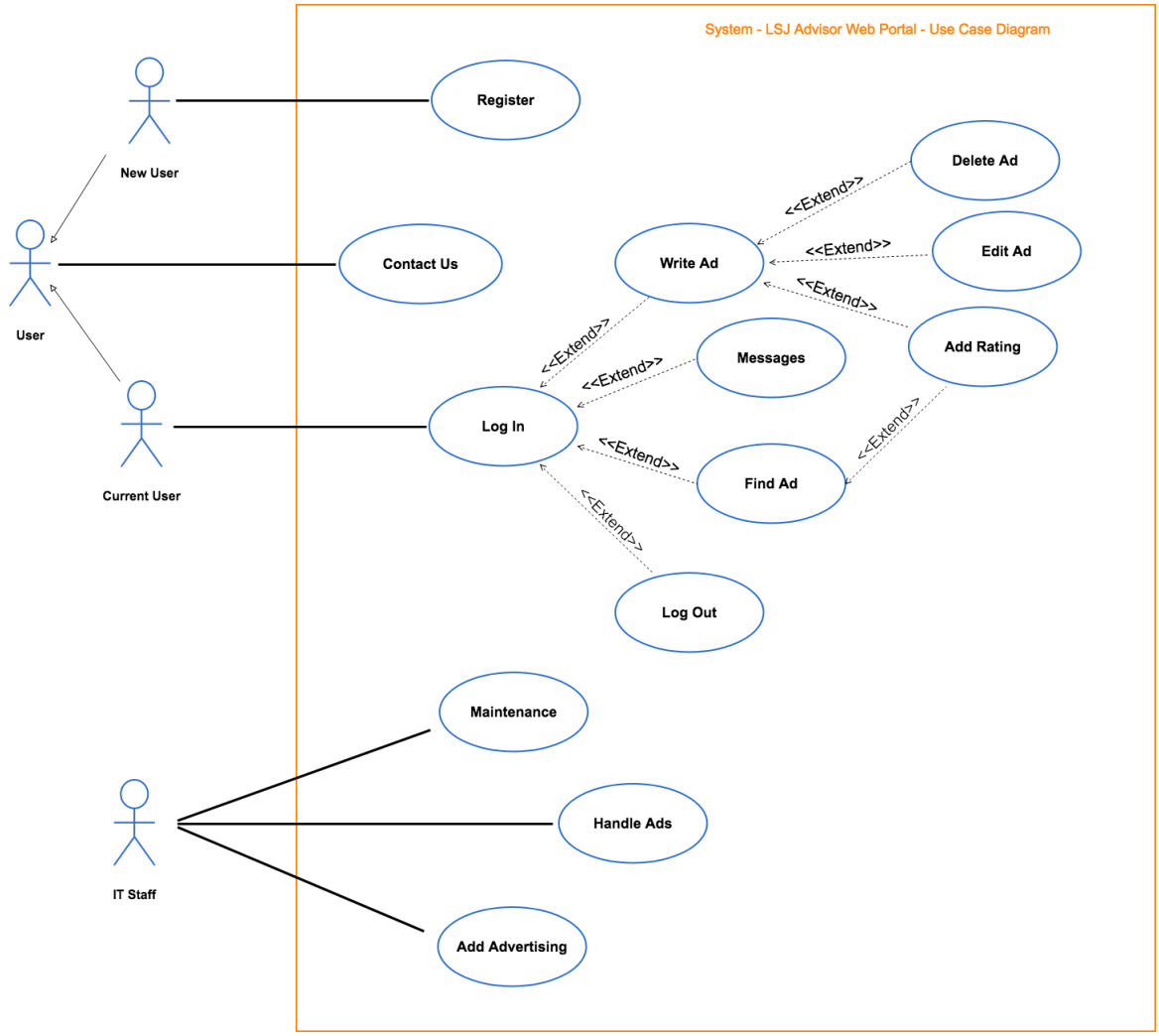


The orange rectangle determinates the system of L.S.J Advisor and separates it from what is outside.



We identified four actors who can interact with the website and used a continuous line to link each of them with the use cases they can participate in. User and IT Staff are the two general actors involved. IT Staff represents all the technicians and support staff members who work on the program for maintenance purposes. User represents actors who are customers of L.S.J Advisor. The arrows coming from the New User and Current User actors and ending at User indicates that the former are specialized actors that inherit from the generic actor. Therefore, the generic User can only access the use case which is designated for him (Contact us), while New User and Current User can participate in all the use cases linked to user and in their own use cases too.

Use Case could have a hierarchy represented by an arrow that indicates the specialization/generalization relationship. Between use cases we can find relationships of inclusion and extension: Inclusion defined when some steps must be necessarily performed while extensions are used when a behaviour is optional and subjected to a condition of activation. Once the Current User is logged in, he can choose (extend arrows) whether to write ads, find ads , send a message or log out. After writing an ad, the user will be able to edit or delete it from the related list of ads. In addition, the user can add a rate after the job is done. The IT Staff has access to the activities "Maintenance", "Add Advertising " and Handle Ads .



4.2 Activity Diagrams

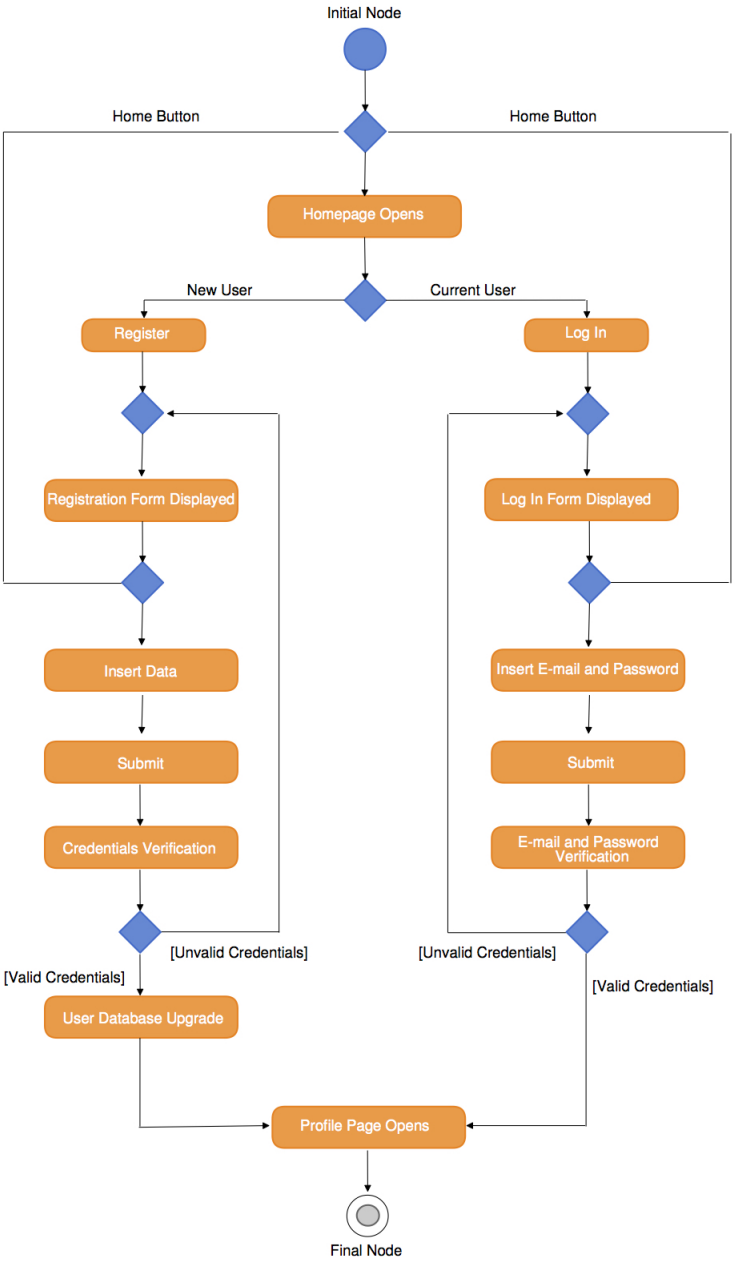
The Activity Diagram is a UML behaviour diagram used for the comprehension of the system behaviour. It shows work flows of activities and actions from the initial node to the end node emphasizing the sequence and conditions of the flow. Because of this dynamic structure, they differ from static Use Case. Different symbols are used in this type of diagram:

- The activities are represented by rounded-rectangles and are connected each other by arrows.
- Starting point of the diagram is represented by an initial node and the finish point by one or more activity final nodes represented by the filled circle with a border.
- Decisions can be represented in this graph through diamonds node and if there are some concurrent activities they can be featured by bars from start (split) or end (join).

Starting from these definitions, the next paragraphs describe the pages of our website through the use of Activity Diagrams.

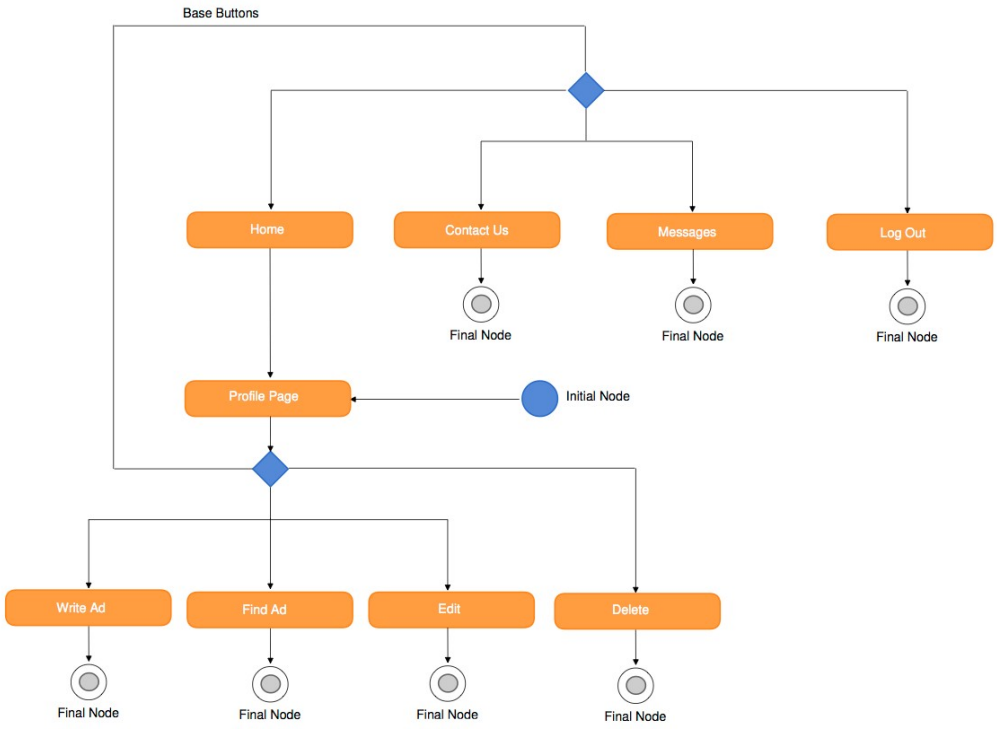
4.2.1 Registration/Login

The first page of the Website is the Welcome Page. It gives the possibility to new Users to register them through the 'Register' option and it has shows the 'Login' option reserved to Users which have already signed up. Besides, without any restrictions, there is also the possibility to contact the staff. The homepage's Base buttons in the taskbar are always available; they allow the user to navigate to "Home" or to "Contact US" at any point. Once a new user has successfully signed up, they becomes a current user and their profile page opens automatically.


Initial Node
Home Button
Home Button
Homepage Opens
New User
Current User
Register
Log In
Registration Form Displayed
Log In Form Displayed
Insert Data
Insert E-mail and Password
Submit
Submit
Credentials Verification
E-mail and Password Verification
[Unvalid Credentials]
[Unvalid Credentials]
[Valid Credentials]
[Valid Credentials]
User Database Upgrade
Profile Page Opens
Final Node

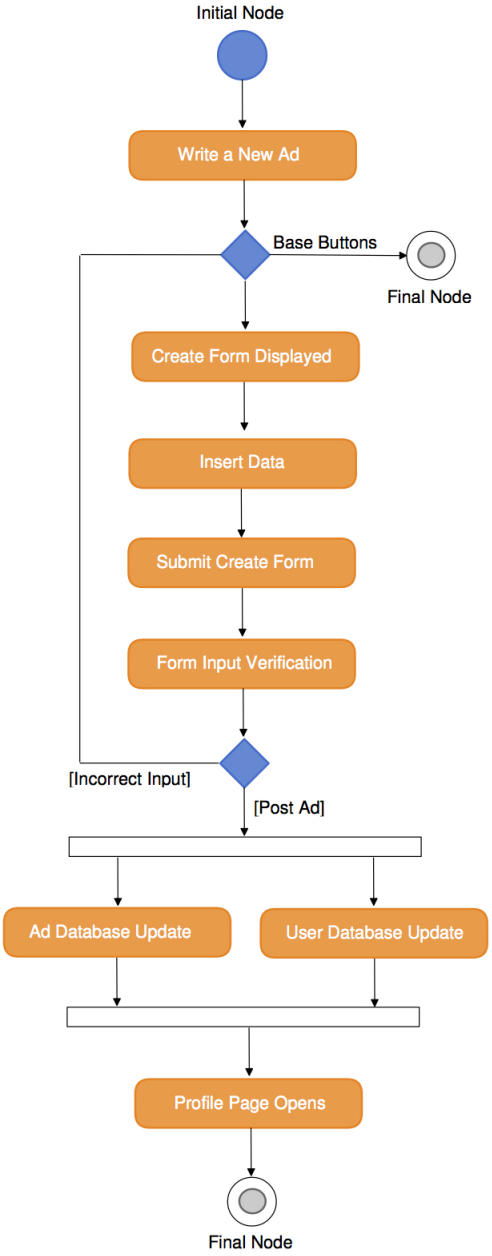
4.2.2 Profile Page

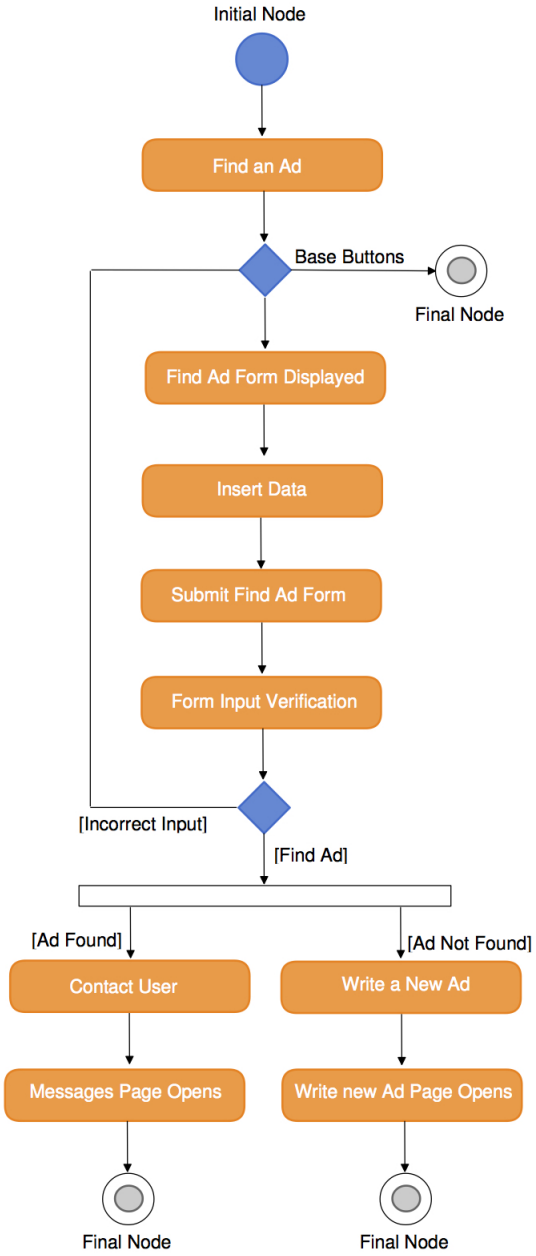
The Profile Page is a central element that is connected to all the different activities that the user can perform, such as write ad, find ad or log out.



4.2.3 Write Ad/Find Ad

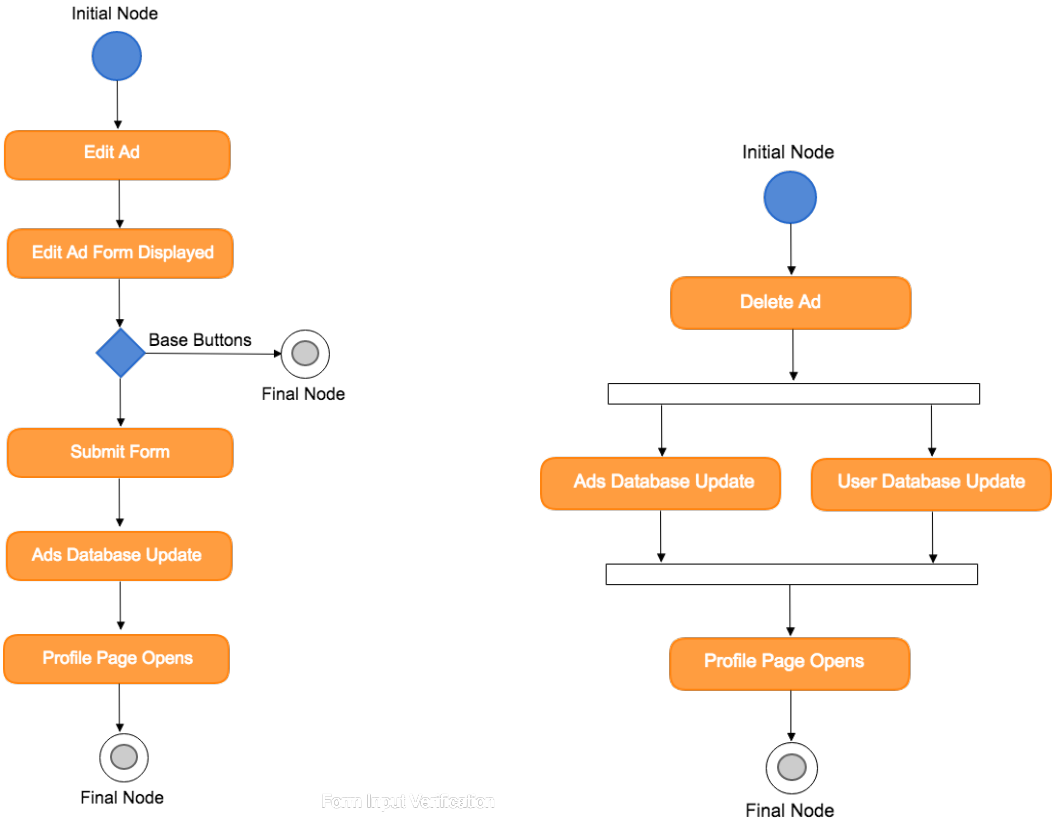
After the access with the correct credentials is done, the Profile Page opens and it allows user to choose if they want write a new ad or find an existent one in the database. After that, the user will be able to insert the requested data in a form and then the ad will be published/showed.





4.2.4 Edit/Delete Ad

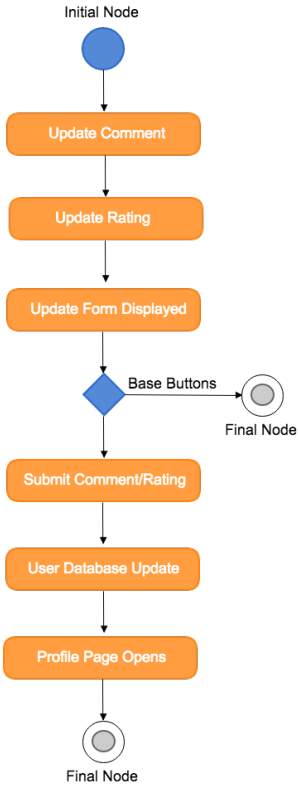
The user who wrote an ad can also edit or delete it.



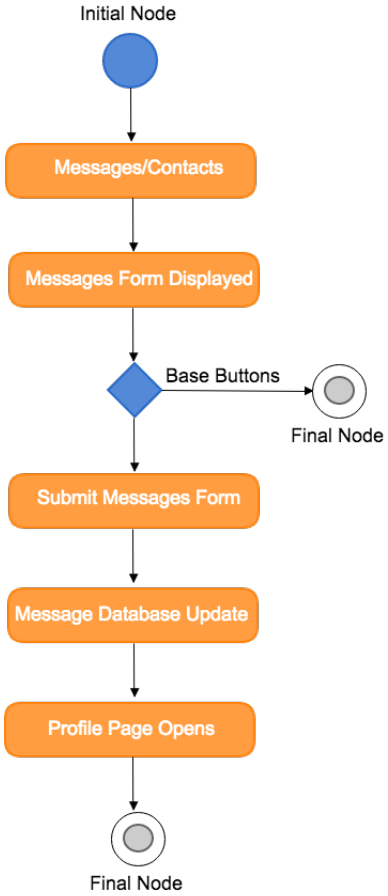
(a) (b)

4.2.5 Add Rating

Users may evaluate the low-skilled job made by the job taker through a comment and a rating system. This rating system consists in assigning a score from 0 to 5 point. Also the job taker can write a comment about the user who required the job.

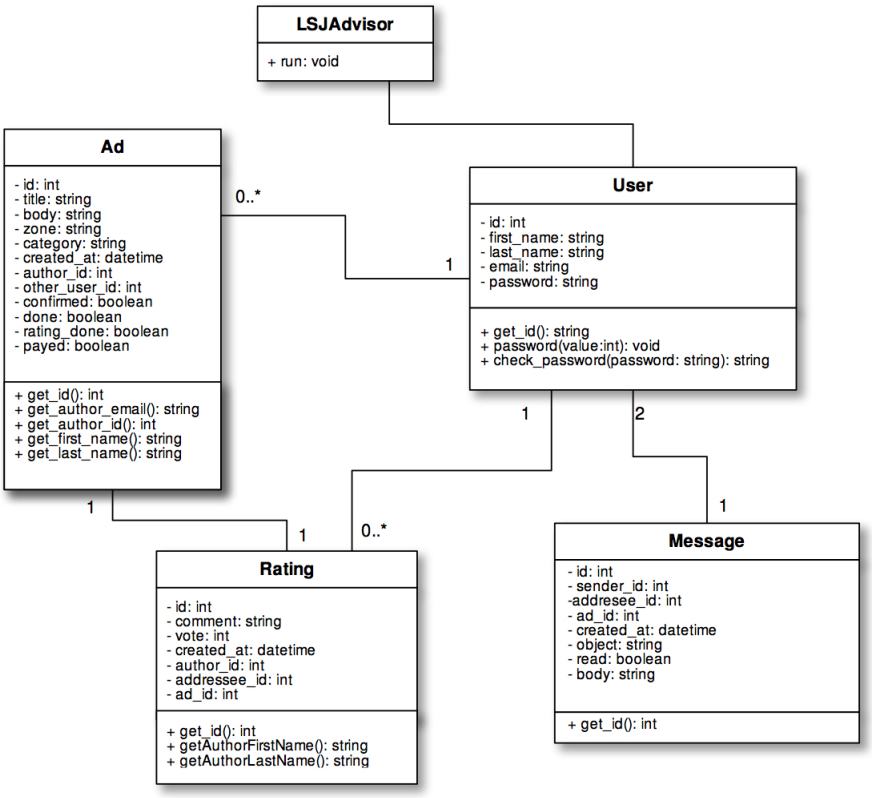


4.2.6 Messages/Contacts



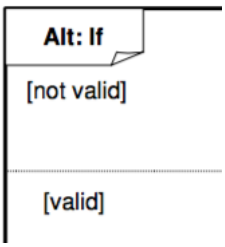
4.3 Class Diagram

The Class Diagram, as the previous diagrams, follows the paradigm of Object Oriented Programming. It is a structure diagram and represents a static model of how the different objects of a system interact. The objects belong to a specific class, which defines their attributes and their general behaviour. Association is the most common relationship between classes. It is represented by a line with multiplicity indexes. These specify numerically the relationships between the objects of the classes. For example, the relationship between User and Ad is one to many, because user can write as many ads as they need.



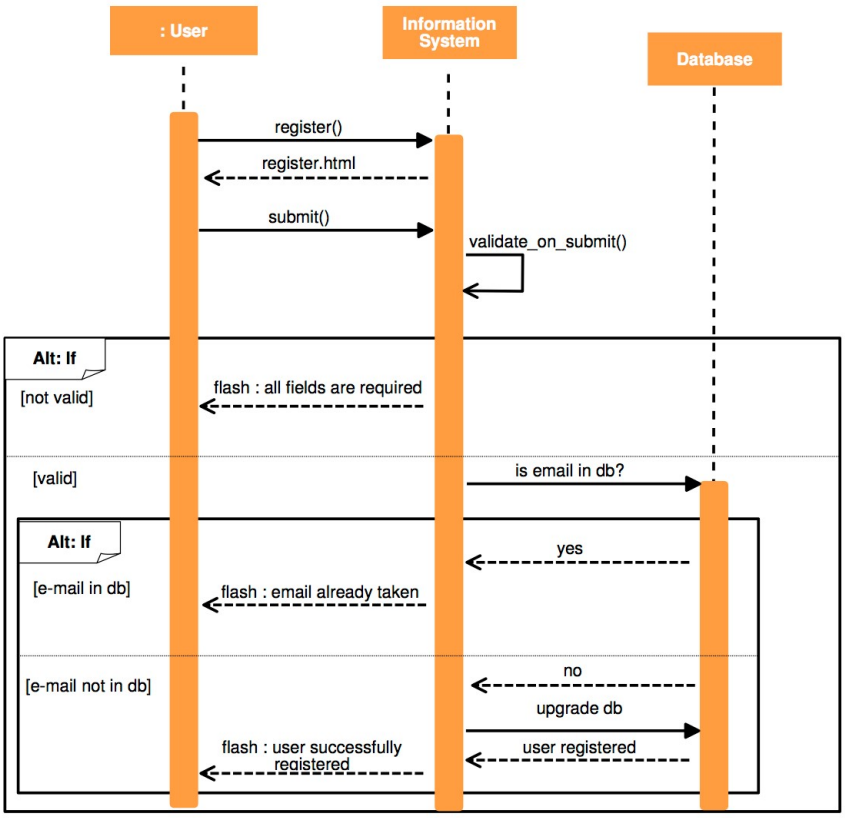
4.4 Sequence Diagrams

A Sequence Diagram is an Interaction Diagram that shows how the characters in each process interact with each other and in which order. The characters involved are either actors or objects. They are located at the top of the diagram and each of them has its own lifeline represented by a dotted line when the character is not active and represented by a narrow rectangle when it is. The diagram has to be read from the top to the bottom, following the horizontal arrows, which describe the exchanging messages from one entity to the other, according to the time flow. To describe more complex interactions, operators have to be used in order to specify which kind of sequence is performed (if, then, else, for, while, etc.).

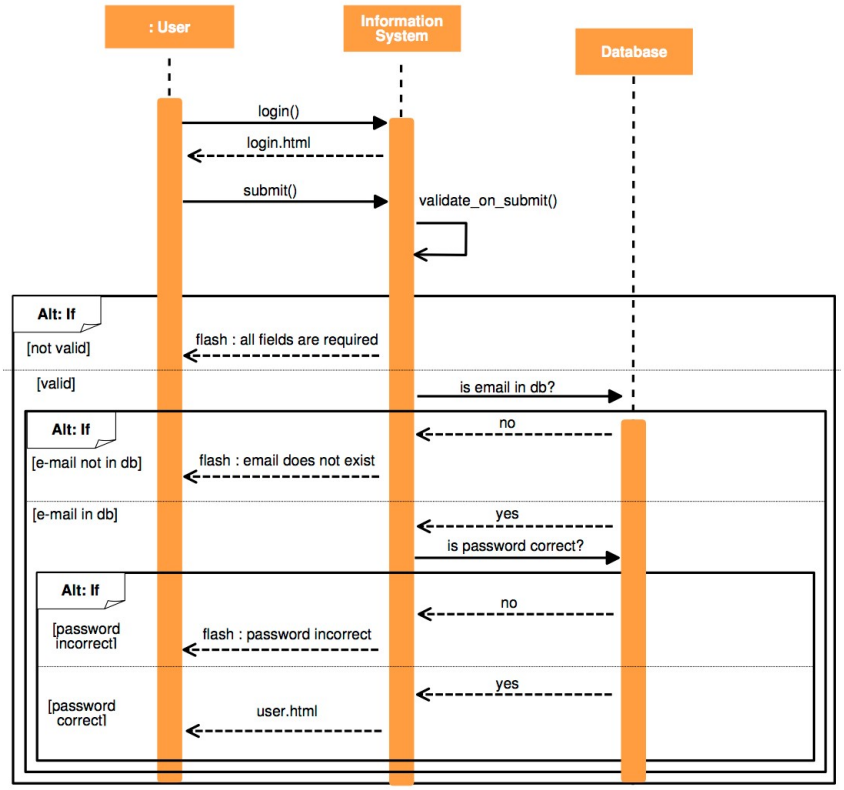


An operator is represented by a bounding box, or frame, that delimits those sub areas of the sequence diagram that have to be executed following its "interaction rules".

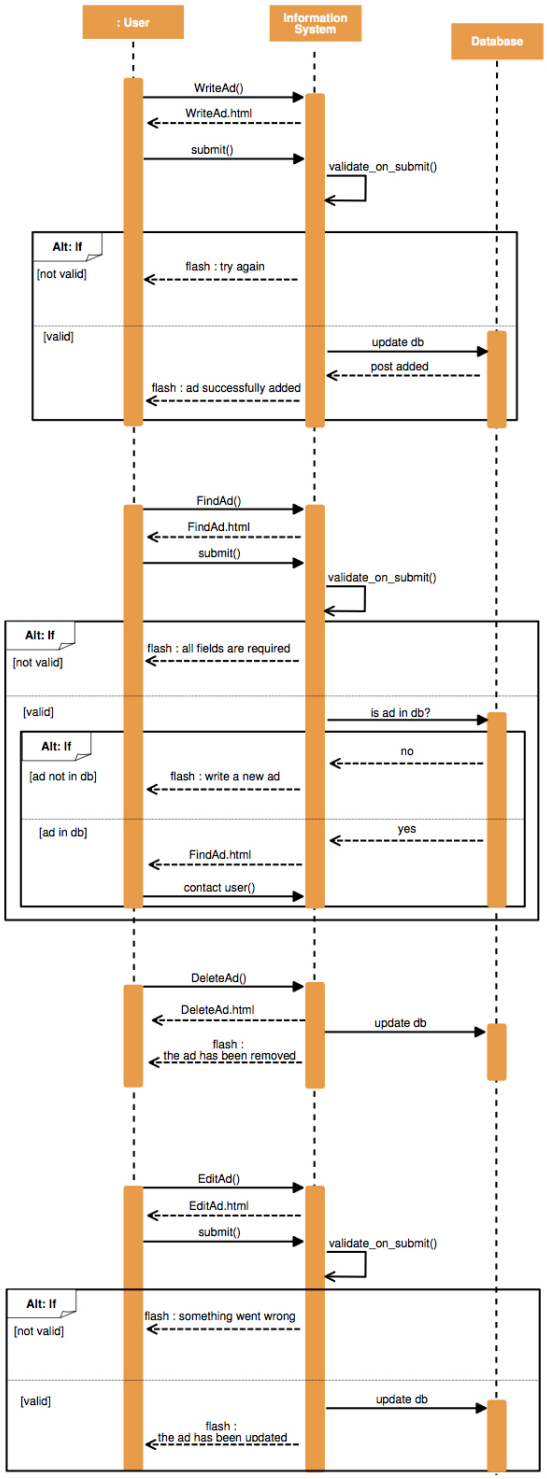
4.5 Registration



4.6 Login


: User
Information System
Database
login()
login.html
submit()
validate_on_submit()
Alt: If
[not valid]
flash : all fields are required
[valid]
is email in db?
Alt: If
no
[e-mail not in db]
flash : email does not exist
[e-mail in db]
yes
is password correct?
Alt: If
no
[password incorrect]
flash : password incorrect
yes
[password correct]
user.html

4.7 Write/Find/Delete/Edit Ad



5 Implementation

The web application was coded in Python using the Flask Framework. We divided the work into two parts. The first one was the static part, with the frontend website programming. The second part was the dynamic one, where we use the Python language to approach the site function with the server-side. For the first part we used client-side languages: HTML, CSS and Bootstrap. For the dynamic part, after learning the basis of Python, we approached the use of some useful tools: Flask, SQLAlchemy and Jinja2. The Python language helps us to build the logic of the web application, Flask to handle active pages and SQLAlchemy to communicate with the server and the database.

5.1 Flask framework

Flask is a micro web application framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frameworks related tools. The main libraries used were: flask, wtforms and werkzeug.security. The first two modules allowed the easy manage of functions and forms. The third one is used to generate a password hash to keep the password safe. Since the browser only wants to read something and make few easy operations, the server has to handle everything concerns databases and variables-handling. This means that building an application through Python required, to work the best way, a distinction between the logic (on the server side), and the GUI (graphic user interface on the client side). Using the render template module it was easy render a template, written in html and javascript, to present it to the client. When needed,into the render_template command we can pass some variables that had to be manipulated and passed to the template. The Jinja2 template engine allows customization of tags, filters and tests. Also, Jinja2 allows the template designer to call functions with arguments on objects. We also use the flask_mail and itsdangerous libraries to implement the email confirmation in the registration phase.

5.2 Database structure

A database stores application data in an organized way. The application then issues queries to retrieve specific portions as they are needed. The most commonly used databases for web applications are those based on the relational model, also called SQL databases in reference to the Structured Query Language they use. To interact with it we used SQLAlchemy, a flask library, that interacts with the user and the database itself to capture and analyze data. It is a fundamental part of the website, because by interacting with the server it's possible to made available the information for users. In our project we created an internal database with four tables.

5.2.1 User table

The User table is the one that collects the information about all the users when they sign up and when they sign in, to verify if the credential used are correct. The code of the class that creates the table User is:

```
class User(db.Model, UserMixin):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    first_name = db.Column(db.String)
    last_name = db.Column(db.String)
    email = db.Column(db.String, nullable=False, unique=True, index=True)
    password_hash = db.Column(db.String, nullable=False)
    confirmed = db.Column(db.Boolean)
```

The User class is a subclass of the database class models.py, meaning that it's a new table. The "id" column is the unique primary key, which holds a unique identifier for each row stored in the table. The "confirmed" column is a boolean useful to check if the user has confirmed his email or not.

5.2.2 Ad table

This table is used to collect all the post written by the user of the table user. The code of "Ad" table is:

```
class Ad(db.Model):
    __tablename__ = 'ads'
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(255))
    body = db.Column(db.Text)
    zone = db.Column(db.String(255))
    category = db.Column(db.String(255))
    created_at = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    author_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    other_user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    confirmed = db.Column(db.Boolean)
    done = db.Column(db.Boolean)
    rating_done_one = db.Column(db.Boolean)
    rating_done_two = db.Column(db.Boolean)
    payed = db.Column(db.Boolean)

    rating = db.relationship('Rating', backref='rating', lazy='dynamic')
    author = db.relationship('User', backref='author', foreign_keys=[author_id])
    other = db.relationship('User', backref='other', foreign_keys=[other_user_id])
```

The "ratings", "author" and "other" parameter however is not an attribute but a back-reference to a relationship with the classes Rating and User. Each ad has an author, another user who answer to the post and the ratings of both users.

5.2.3 Message table

The Message table is used to implement a form of messaging between users. The code is:

```
class Message(db.Model):
    __tablename__ = 'messages'
    id = db.Column(db.Integer, primary_key=True)
    sender_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    addressee_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    ad_id = db.Column(db.Integer, db.ForeignKey('ads.id'))
    created_at = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    object = db.Column(db.String(255))
    read = db.Column(db.Boolean)
    body = db.Column(db.Text)

    sender = db.relationship('User', backref='sent_messages', foreign_keys=[sender_id])
    addressee = db.relationship('User', backref='received_messages', foreign_keys=[addressee_id])
    ad = db.relationship('Ad', backref='ad', foreign_keys=[ad_id])
```

5.3 Forms

When using Flask-WTF, each web form is represented by a class that inherits from class Form. The class defines the list of fields in the form, each represented by an object. Each field object can have one or more validators attached; validators are functions that check whether the input submitted by the user is valid. In our project there are seven forms: RegistrationForm, LoginForm, EditProfileForm, WriteForm, FindForm, EditForm, WriteMessageForm and WriteRatingForm. The fields and validators are imported directly from the WTForms package.

```python
class RegistrationForm(Form):
    first_name = StringField('First name', validators=[DataRequired("Please enter your first name")])
    last_name = StringField('Last name', validators=[DataRequired("Please enter your last name ")])
    email = StringField('Email', validators=[DataRequired("Please enter your Email address"), Length(min=4)])
    password = PasswordField('Password', validators=[DataRequired("Please enter a password"), Length(min=6)])
    password2 = PasswordField('Repeat password',
                              validators=[DataRequired(),
                                          EqualTo('password', message="Please check both passwords match")])
    submit = SubmitField('Register')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is not None:
            raise ValidationError('Email already exists')
```

```python
class LoginForm(Form):
    email = StringField('Email', validators=[DataRequired("Please enter your Email address")])
    password = PasswordField('Password', validators=[DataRequired("Please enter your password")])
    submit = SubmitField('Login')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('Email does not exist')
```

```python
class EditProfileForm(Form):
    first_name = StringField('First name', validators=[DataRequired("Please enter your first name")])
    last_name = StringField('Last name', validators=[DataRequired("Please enter your last name ")])
    password_last = PasswordField('Last password',
                                  validators=[DataRequired("Please enter your last password"), Length(min=6)])
    password = PasswordField('New password', validators=[DataRequired("Please enter a new password"), Length(min=6)])
    password2 = PasswordField('Repeat password',
                              validators=[DataRequired(),
                                          EqualTo('password', message="Please check both passwords match")])
    submit = SubmitField('Edit')
```

```python
class WriteForm(Form):

    categories_not_ordered = ['Houseworks', 'Children', 'School', 'Animals', 'Post Office', 'Supermarket']
    zones_not_ordered = ['Crocetta', 'Santa Rita', 'Cenisia', 'San Paolo', 'Cit Turin',
                         'Vanchiglia', 'Barriera Milano', 'Quadrilatero', 'Centro']

    categories_not_ordered.sort()
    zones_not_ordered.sort()

    categories = [(c, c) for c in categories_not_ordered]
    zones = [(a, a) for a in zones_not_ordered]

    title = RadioField(choices=[('I need', 'I need'), ('I offer', 'I offer')],
                       validators=[DataRequired("Please choose a type of ad")])
    body = TextAreaField("Description", validators=[DataRequired("Please enter a short description")])
    category = SelectField(label="Category", choices=categories)
    zone = SelectField(label="Zone", choices=zones)
    submit = SubmitField('Add')
```

```
class FindForm(Form):

    categories_not_ordered = ['Houseworks', 'Children', 'School', 'Animals', 'Post Office', 'Supermarket']
    zones_not_ordered = ['Crocetta', 'Santa Rita', 'Cenisia', 'San Paolo', 'Cit Turin',
                         'Vanchiglia', 'Barriera Milano', 'Quadrilatero', 'Centro']

    categories_not_ordered.sort()
    zones_not_ordered.sort()

    categories = [(c, c) for c in categories_not_ordered]
    zones = [(a, a) for a in zones_not_ordered]

    title = RadioField(choices=[('I offer', 'I need'), ('I need', 'I offer')],
                       validators=[DataRequired("Please choose a type of ad")])
    category = SelectField(label="Category", choices=categories, validators=[DataRequired()])
    zone = SelectField(label="Zone", choices=zones, validators=[DataRequired()])
    submit = SubmitField('Find')
```

```
class EditForm(Form):

    categories_not_ordered = ['Houseworks', 'Children', 'School', 'Animals', 'Post Office', 'Supermarket']
    zones_not_ordered = ['Crocetta', 'Santa Rita', 'Cenisia', 'San Paolo', 'Cit Turin',
                         'Vanchiglia', 'Barriera Milano', 'Quadrilatero', 'Centro']

    categories_not_ordered.sort()
    zones_not_ordered.sort()

    categories = [(c, c) for c in categories_not_ordered]
    zones = [(a, a) for a in zones_not_ordered]

    title = RadioField(choices=[('I need', 'I need'), ('I offer', 'I offer')],
                       validators=[DataRequired("Please choose a type of ad")])
    body = TextAreaField("Description", validators=[DataRequired("Please enter a short description")])
    category = SelectField(label="Category", choices=categories, validators=[DataRequired()])
    zone = SelectField(label="Zone", choices=zones, validators=[DataRequired()])
    submit = SubmitField('Edit')
```

```
class WriteMessage(Form):
    object = StringField("Object", validators=[DataRequired("Please enter an object")])
    body = TextAreaField("Message", validators=[DataRequired("Please enter a message")])
    submit = SubmitField('Send')
```

```
class WriteRating(Form):

    votes_not_ordered = ['1', '2', '3', '4', '5']
    votes_not_ordered.sort()
    votes = [(c, c) for c in votes_not_ordered]

    comment = TextAreaField("Comment")
    vote = SelectField(label="Vote", choices=votes, validators=[DataRequired("Please enter a vote")])
    submit = SubmitField('Add rating')
```

In the Class definition we only put inside the attributes of the class, and we write all the others methods in the "main" class, called views.py.

5.4 Main functions

In this section we will present the main functions of our website. The following images are from views.py page, but all the interconnections with the database are possible with the files form.py and model.py. All the files are linked in the file __init__.py. Because of this, in the code below we will see lot of methods coming from that page.

5.4.1 Registration,log in, log out

This is the code that allow a user to do the signup. In the page registration.html there is a form where the user can insert his credentials. After checking all the data, if the username is not already in the database and the email is validating, the new user is correctly insert in the database and he is automatically logged in.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        u = User(first_name=form.first_name.data, last_name=form.last_name.data,
                 email=form.email.data, confirmed=False)
        u.password = form.password.data
        db.session.add(u)
        db.session.commit()
        token = generate_confirmation_token(u.email)
        confirm_url = url_for('confirm_email', token=token, _external=True)
        html = render_template('email.html', confirm_url=confirm_url)
        subject = "Please confirm your email"
        send_email(u.email, subject, html)
        flash('A confirmation email has been sent via email', 'success')
        return redirect(url_for('index'))
    return render_template('register.html', form=form)
```

After the user is registered, he will have received an email for the confirmation of his registration. We create also an email 'lsjadvisor@gmail.com'. After the user click on the bottom 'register' the server sends an email for account confirmation.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        u = get_user(form.email.data)
        if not u.confirmed:
            flash('Please confirm your email address before login', 'danger')
            return redirect(url_for('login'))
        else:
            if u.check_password(form.password.data):
                login_user(u)
                flash('User logged in!', 'success')
                return redirect(url_for('user'))
            else:
                flash('Incorrect password!', 'danger')
    return render_template('login.html', form=form)
```

It's also possible for a user do the logout.

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('User logged out!', 'success')
    return redirect(url_for('index'))
```

5.4.2 Write ad, Find ad,Remove ad

Another tool offered by our website is to write and find ads, that allows the interconnection between users. If the user does not find what he needs, he can add a post on the website. The ads posted but not yet completed, can be edited and removed.

```
@app.route('/writeAd', methods=['GET', 'POST'])
def writeAd():
    form = WriteForm()
    if form.validate_on_submit():
        ad = Ad(title=form.title.data, category=form.category.data, body=form.body.data,
                zone=form.zone.data, author=current_user,
                other_user_id=None, done=False, rating_done_one=False,
                rating_done_two=False, payed=False, confirmed=False)
        db.session.add(ad)
        db.session.commit()
        flash('Ad successfully added!', 'success')
    ads = Ad.query.order_by(Ad.created_at.desc()).all()
    return render_template('writeAd.html', form=form, ads=ads)
```

```
@app.route('/findAd', methods=['GET', 'POST'])
@login_required
def findAd():
    form = FindForm()
    ads = []
    if form.validate_on_submit():
        ads = Ad.query.filter_by(title=form.title.data, zone=form.zone.data,
                                 category=form.category.data, done=False,
                                 confirmed=False).order_by(Ad.created_at.desc()).all()
        return render_template('findAd.html', form=form, ads=ads)
    return render_template('findAd.html', form=form, ads=ads)
```

```
@app.route('/editAd/<int:id>', methods=['GET', 'POST'])
@login_required
def editAd(id):
    ad = Ad.query.get_or_404(id)
    if current_user.id != ad.author_id:
        abort(403)
    form = EditForm()
    if form.validate_on_submit():
        ad.title = form.title.data
        ad.body = form.body.data
        ad.category = form.category.data
        ad.zone = form.zone.data
        db.session.add(ad)
        db.session.commit()
        flash('The ad has been updated', 'success')
        return redirect(url_for('writeAd', id=id))
    form.title.data = ad.title
    form.body.data = ad.body
    form.category.data = ad.category
    form.zone.data = ad.zone
    return render_template('editAd.html', form=form, ad=ad)
```

```python
@app.route('/deleteAd/<int:id>', methods=['GET', 'POST'])
@login_required
def deleteAd(id):
    ad = Ad.query.get_or_404(id)
    if current_user.id != ad.author_id:
        abort(403)
    db.session.delete(ad)
    db.session.commit()
    flash('The ad has been removed', 'success')
    return redirect(url_for('user'))
```

5.4.3 Write message, Add rating

When the user finds an ad to which he is interested, he can contact the author of the post to give and receive more information. If the job is confirmed, done and payed, users can add a rating about.

```python
@app.route('/writeMessage/<int:id>', methods=['GET', 'POST'])
@app.route('/writeMessage/<int:id>/<int:ad_id>', methods=['GET', 'POST'])
@login_required
def writeMessage(id, ad_id):
    form = WriteMessage()
    if form.validate_on_submit():
        addressee = User.query.filter_by(id=id).first()
        message = MessageChat(ad_id=ad_id, sender=current_user, body=form.body.data,
                              addressee=addressee, read=False, object=form.object.data)
        db.session.add(message)
        db.session.commit()
        flash('Message successfully sent!', 'success')
        return redirect(url_for('writeMessage', id=id, ad_id=ad_id))
    return render_template('writeMessage.html', form=form)
```

```python
@app.route('/addRatingOne/<int:ad_id>', methods=['GET', 'POST'])
@login_required
def addRatingOne(ad_id):
    form = WriteRating()
    if form.validate_on_submit():
        ad = Ad.query.get_or_404(ad_id)
        rating = Rating(ad_id=ad_id, author_id=current_user.id, addressee_id=ad.author_id,
                        comment=form.comment.data, vote=form.vote.data)
        ad.rating_done_one = True
        db.session.add(rating)
        db.session.add(ad)
        db.session.commit()
        flash('Rating successfully added!', 'success')
    return render_template('writeRating.html', form=form)

@app.route('/addRatingTwo/<int:ad_id>', methods=['GET', 'POST'])
@login_required
def addRatingTwo(ad_id):
    form = WriteRating()
    if form.validate_on_submit():
        ad = Ad.query.get_or_404(ad_id)
        rating = Rating(ad_id=ad_id, author_id=current_user.id, addressee_id=ad.other_user_id,
                        comment=form.comment.data, vote=form.vote.data)
        ad.rating_done_two = True
        db.session.add(rating)
        db.session.add(ad)
        db.session.commit()
        flash('Rating successfully added!', 'success')
    return render_template('writeRating.html', form=form)
```

All the project is available on GitHub: https://github.com/rebeccapelaca/LSJAdvisor

6 Conclusion

6.1 The idea and its implementation

Since the beginning we put a lot of effort into finding the right idea for our project. In several informal meetings, we discussed about several possible ideas. In a relative short time we figured out that we did not want to develop the classic job market place portal, but instead do something original. After hours of brainstorming "LSJ" was created. The project concept was chosen, however during the development many functions and processes were proposed, discussed, implemented or rejected. On the one hand we were really excited to create a great platform able to be competitive in the real world, but on the other hand we had to face the limits of our competences. However, the result we have reached make us feel very proud of ourselves, with the perspective of great improvement of LSJ Advisor to be achieved in the future! Thus, how has LSJ Advisor been developed? Starting from the forming process of the team, we tried to get a general idea about the competences of each team member. Some members have a background in industrial engineering, while the others come for a more management oriented background. As a consequence, most of the applied methods were learned through the lab courses and the recommended tutorials (Codeacademy, Flask megatutorial). Then, depending on knowledge, one or more persons worked on specific areas and got support from the rest of the group if necessary.

6.2 Future development of LSJ Advisor

LSJ Advisor is a simple and immediate website. The intuitive and easy navigation allows the user to join LSJAdvisor without any learning or adapting. Looking at the future, a potential upgrade would be to add a secure payment tool in order to manage the exchange of money. We could also provide a better website design, with a great graphics. Our intent for the future is to reserve more space for advertising, involving supermarkets and schools. Another change could be to provide a mobile application of LSJ Advisor. It would be more convenient for users to use our service while they are on the move. We could also provide a Google Maps integration, which would show the most interesting ads.

7 References

- Claudio Giovanni Demartini, **"Information Systems" Course Slides** , 2016
- Kenneth A. Lambert, **"Fundamentals of Python: First Programs"**, 2012
- Miguel Grinberg, **"Flask Web Development"**, 2014
- Flask (A Python microframework) – Pocoo, **Flask Extensions**, http://flask.pocoo.org/extensions/
- TutorialsPoint, **Flask Tutorial**, http://www.tutorialspoint.com/flask/
- https://www.w3schools.com/html/
- https://pythonspot.com/en/flask-web-app-with-python/
- https://www.codecademy.com/learn/all
- https://github.com/