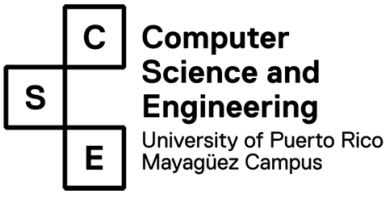
University of Puerto Rico Mayaguez
Department of computer Science and Engineering
INSO 4101 Introduction to Software Engineering



Homework 2:

Domain: App development/Bounty Board

Anthony Zapata, Ebdiel Roman Feliciano, Javier Maldonado, Songiemar Garcia

March 6th, 2020

Prof. Marko Schuetz Shmuck

I. Chapter 1 in SE-V3

A. Domain Entities

1. Simple Entities

a) Rankings

1) A user score based on the reviews of the other party. It has two independant rankings, contactor and acceptor.

b) User Reviews

1) The users, both contractors and acceptors give a score based on the quality of the task done, or fairness or pay of the request.

c) Specifications

1) The requirements for the tasks in order to be considered complete.

d) Moderator:

1) In charged of making any changes to any data in the software.

i) Can modify/remove/add categories

ii) Can remove users

iii) Can remove Jobs

iv) Can remove reviews

2. Composite Entities

a) Users: Are people using our software.

1) Profile: A profile is a sensitive document that contains:

i) Full name: Name and last name

ii) Age: What age is the user.

iii) Gender: Male, female or other.

iv) Location: Place of residence or place of task.

v) Cell phone number: Personal number for contacting between users.

vi) Email: Additional contact info.

vii) Social media integrations: facebook, Whatsapp, etc.

viii) Brief description on what they specialize in.

ix) Ranking: Total ranking in comparison to other users

x) Resume: Professional experience document.

b) Category:

1) Jobs: Can have any specifications to it.

c) Bounty Hunt

1) Request: the act of placing a post for a task so others can apply for it.

2) Applicant: One of multiple possible users to contract for a job, based on how high the user's rank is hto more likely they are to be accepted for that job.

3) Confirmation process: The act choosing an applicant based on the personal criteria.

4) Completion: When a task is completed, the users undergo the review process and exchange process.

5) Review process: A process where you evaluate the other party based on a set criteria. This influences their rank.

6) Monetary Exchange

i) Service Coins: It is a digital currency created by the developers.

ii) The conversion of real world money into digital currency solely to streamline the payment process.

3. Attributes and Values

a) Simple Ranking Score

1) Based on a leaderboard, who ever has smaller number means that the higher they are in the leaderboard.

2) On the scale of 1➡∞.

b) Simple Reviews

1) Based on a score of 0 ➡10. Decimals included.

2) Classification based in the quality of the completed task or the pay/fairness of the contactor.

c) Composite User

1) Users can be nearby or not.
2) Users can have a high rank or not.
3) Users can be easily accessible or not by their contact information.
4) User can be qualified for a job or not.

d) Composite Bounty Hunt

1) Can be completed or not.
2) Currency exchange is handled with Service Coins.

B. Domain Functions

a. Post job - Post : Server ➡ Jobs ➡ Job ➡ Add >< Server
b. Accept job - Accept : Server ➡ Jobs ➡ Job ➡ Take >< Server
c. Create user - Create : Server ➡ Users ➡ User >< Server
d. Update user - Update : Users ➡ User ➡ Update >< User
e. Review user - Review : Users ➡ User ➡ Review >< User
f. Report user - Report : Users ➡ User ➡ Report >< User
g. Search Job - Search : Server ➡ Location ➡ Jobs
h. Check user profile- Search: Server ➡ Users ➡ User ➡ Info
i. Payments- Pay: User ➡ Info >< Transaction
j. Notify Applicants - Notify : Server ➡ Users ➡ Applicants ➡ Notify
k. Exchange Currency : Trade ➡ Users ➡ User1 User2 >< Transaction

C. Domain Events

a. Job posted : User successfully posted job to the server.
b. Job accepted : User successfully accepted job from the server.
c. Job look up : User successfully found jobs related to its location.
d. User review : The user completed the review section after the job was done.
e. Transaction : Successful transaction between two users.

D. Domain Behaviors

- Users log in with their own credentials to look for and/or post tasks for others.
- Applicant surfing through the server looking for a job.

E. Domain Requirements
   - Searching for tasks: the application must allow users to search for tasks based on keywords or categories.
   - See detailed task information: The application must show the user detailed information on the task the are viewing including the name of the user who submitted the task, the reward for completing the task and due date for the task to be completed upon accepting it.
   - See user reviews: The application must let users see other user's reviews of both the submitter of the task and those who are offering to complete said task.
   - Show tasks by proximity: The application must show users tasks that are close to their proximity so as to not have situations where a user accepts a task that they are unable to complete due to being too far from the location.

F. Interface Requirements
   - The application shall display a webpage to the user with multiple UI components:
     - o A Search Box: Let's user search with specific keywords
     - o A dropdown list By Category: the list will display broad categories that the task can be grouped into to narrow the list of tasks show to the user.
     - o A Table of Cards: The application will show the tasks as an organized list of cards that resize according to the size of the screen of the user's device.
   - The Main page shall display the newest posted tasks to the user

- When the user interacts with any of the tasks cards the Page should expand the card in a modal popup and present the user with information like:
    - Task.
    - User that submitted the task.
    - Reward for completion of the task.
    - More detailed explanation of the task.
    - A button to accept the task.
    - A button to get in contact with the User that posted the task for more information.
    - A small button to close the card without accepting.
    - Review of the user that submitted the task.

G. Machine Requirements

1. Performance Requirements
    a) Server must be efficient, able to respond in the shortest amount of time and handle multiple requests at once.
    b) Server must be able to handle a large amount of users at once. Load will be a factor that will affect the function of the system. This effects must be countered as soon as they appear.
    c) Application must be light enough (as low load as possible) in order to function in a variety of devices such as smartphones and regular computers.
    d) Application distribution is affected by package size (Final downloadable size file). Maintain as small as possible.
2. Dependability Requirements

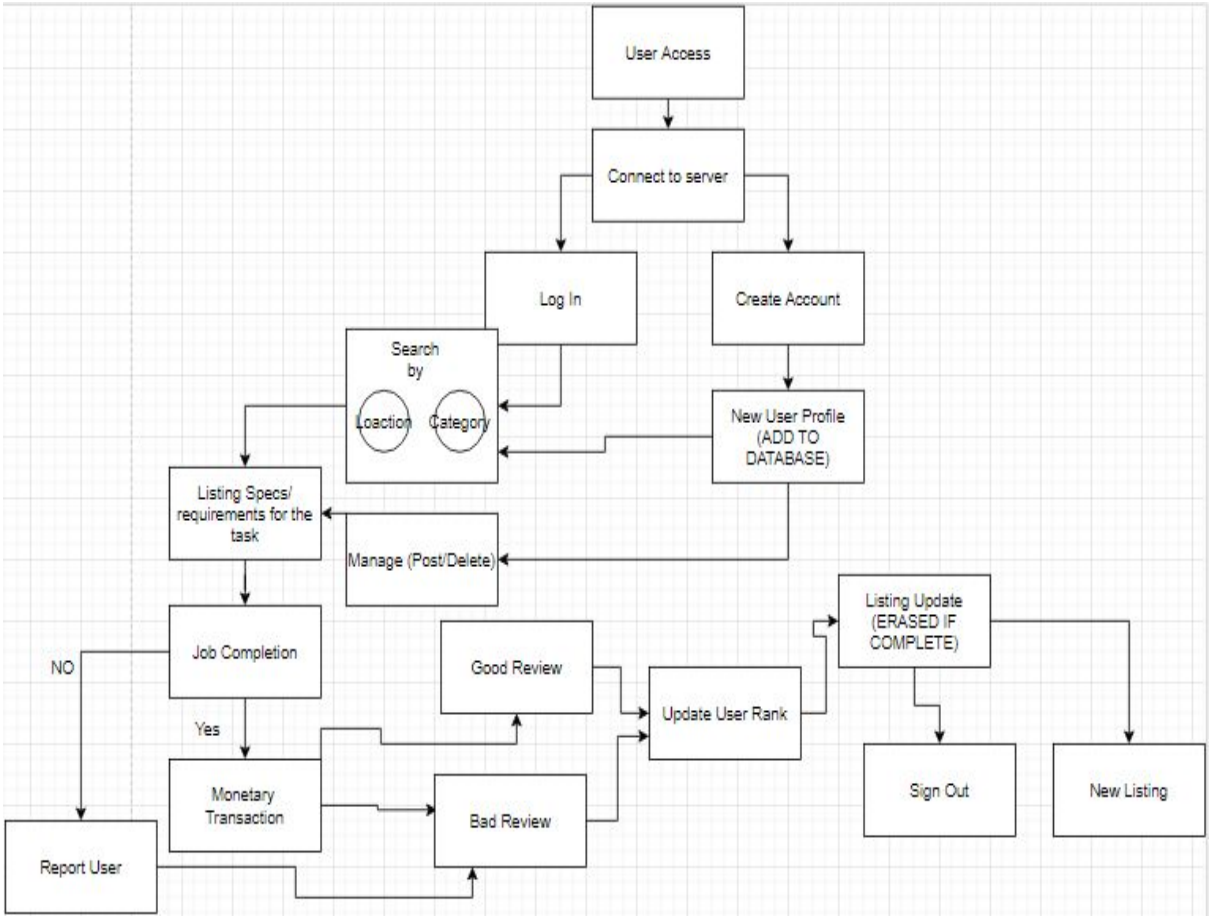
a) Servers must be checked bi-weekly (Hardware and Software side) in order to maintain a reliable system operating.
b) In case of external problems, server must be re-initialized and re-establish its main service in less than 10 minutes. Any fatal errors shall be notified to the end user.
c) Security is the main priority of our software. Personal information will not be revealed to any other user unless the initial user permits it.
d) App and backend are two separate entities which communicate with each other via packages. Any information transaccion will be encrypted and decrypted in the back end, making our software hard to modify by 3rd parties.

3. Maintenance Requirements
    a) Every version of the app will be stored for documentation purposes. All user information shall be stored in a central database. This database shall keep ALL users, independent of activity.
    b) Our application endorses physical user-to-user interaction which may introduce us in liability related situations. Therefore, all policies must remain updated as time goes along. All users MUST be notified of any policy changes in the application functionality.
    c) All user data WILL be maintained safe and only designated personnel will be able to access such information (for maintenance purposes).
4. Platform Requirements
    a) System must be implemented in native mobile app platforms and web browser.
5. Documentation Requirements
    a) User's guide for the usage of the application/web services.
    b) Maintenance Log Book

c) Versions Log Book with changes and updates described

d) Policy

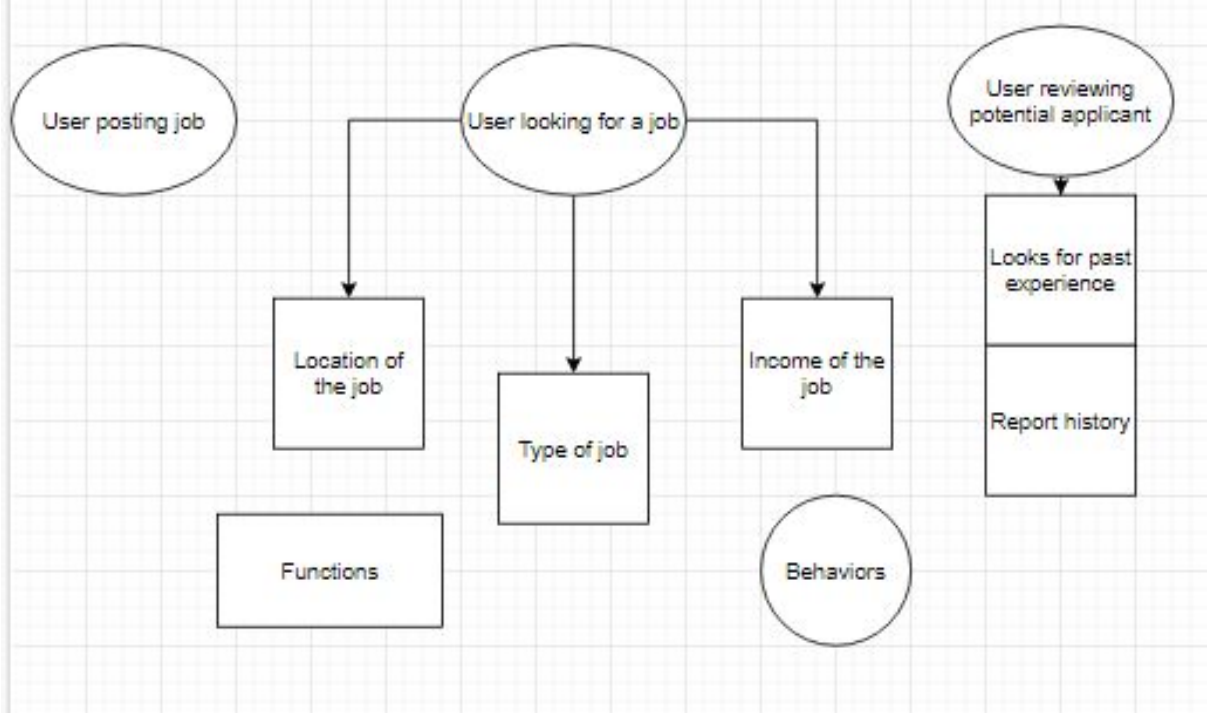
H. Software Architecture Design



I. Software Component Design

1. User information will be stored as “User” type in a Circular Doubly Linked List or HashMap Data Structure.

2. Posts “Jobs” listings will be stored as “Post” type in a Circular Doubly Linked List or HashMap Data Structure.



II. Chapter 2 in SE-V3

A. Informative Domain Development Documents.

I. Name, Place, Date

A. Name:

1. Bounty Board

B. Place:

1. University of Puerto Rico, Mayaguez District

C. Date:

1. Starting Date: February 12th, 2020
2. Estimated Goal Date: May 1st, 2020

II. Partners

A. Clients:

1. For the purpose of this project we are the clients of our domain. As we noticed that there is no specified client, therefore we, the developers, are in charge of identifying a problem to solve, what goes into describing our domain, prescribing our own

specifications, a course of action to pursue in the development of our domain and designing said software related to our domain. If we were to choose a client, it would be based on who is the target demographic such as, professional personnel, unemployed personel, and the requestor.

B. Developers:

1. We are a proud team of four, and we are based in Mayaguez, currently as students in the University of Puerto Rico.
2. Our team consists of:
    a) Anthony Zapata
    b) Ebdiel Roman
    c) Jean Lugo
    d) Kevin Burgos
    a) Javier Maldonado
    b) Songiemar Garcia
    c) Carlos Machin
3. We are trying create an easy and accessible way for users to post jobs, tasks, requests, assignments, errands or chores to other users who are willing to accomplish these for a pre-established monetary value.

III. Current Situation

A. The current situation is that a large amount of the population is either too busy, too lazy or incapable of getting small tasks done and that these tasks could be accomplished by others who are more willing to do them and have the availability to do so provided, of course, they get compensated for their work.

IV. Needs and Ideas

A. Needs:

1. People are in need of assistance of others in order for them complete a task, they cannot or don’t want to complete on their own for whatever reason.
2. In our day to day lives time is a valuable thing, and some people do not have enough time to complete shores, tasks, jobs.
3. A necessity to find a person to do a certain task on demand can be very difficult, specially if you have no contacts or connections to someone who can complete the job.
4. People are also in search for quick money, and are available for short term jobs to gain some.
5. Others simply may not be physically able to do certain task and need to outsource the help. Thats where Bounty Board comes in.

B. Ideas:

1. The idea of this software is to present a user to user job/aid opportunity. Users that may require a certain task to be completed by someone will have the ability to post this task as a "Bounty Hunt" in exchange for monetary rewards. This software provides a profile based system for its users to identify other users that are willing to fulfill this task.
2. Create a web app that displays a board of tasks where users can then browse said tasks and decide if they would like to complete it. The app would also allow users to post their own tasks so that others can accept them.
3. Implementing this software so you can solve the need of needing a helping hand.

V. Concept and Facilities

A. Concepts - The following solution concepts were selected for the app which are the following

a. Provide the user the ability to post to the server.

b. Provide the user the ability to accept jobs from the server.

c. Present a user friendly GUI that will contain the jobs requested from a close enough location.

B. Facilities - It's essential to use the adequate software to complete our project. PostgreSQL will be used to handle the database and server side of the application. For the front-end JavaScript and Angular framework were the language and framework of choice. To keep track of work schedule Scala, ClickUp and other software were chosen.

VI. Scope and Span

A. Scope:

1. The scope of our domain, is generally in software development, specifically a piece of software that can be used by the public. Therefore, l we will produce a piece of software that provides a platform for users to, place request for one time job, and to provide a safe and easy way to contract workers and to make monetary gain or service for the domain holders and the users.

B. Span:

1. The span of our domain, is to facilitate the users of our piece of software to place requests, removes request, accept request form others, place monetary incentive, check expertise of acceptor, for a task that needs to be done. In addition to those functions, you could check by GPSso you could find jobs near your point of origins. You can link your preferred method of payment of choice to provide an accessible way of money transfers

between users. In addition, a rank system would be in place to see who are the most qualified for the job, and the requestor can choose between all of the applicants. All of this would be implemented using REST API, and Scala programming language. In order to provide a platform, for the users to find or create jobs from the necessity of the requestor.

VII. Assumptions and Dependencies

A. Assumptions:

1. It is assumed that the developers have the knowledge and skill to work with the scala programming language, databases, Rest APIs and UX design to create a modern and easy to use web application.
2. It is also assumed that we will have the resources needed to run our servers.

B. Dependencies:

1. Both the quality and efficiency of the application is dependant on this as well as the speed and pace in which the application is completed from design to production.

VIII. Implicit/ Derivative Goal

A. Our goals:

1. Create an easy and accessible way for users to present jobs, tasks, requests, assignments, errands or chores to other users.
2. Established a platform that allows every day person to get some additional income.
3. Ensure user / contractor's satisfaction.
4. Provide a side income for users.
5. Provide easy access for workers to find new tasks to complete (includes private companies).

6. Provide an easy method for people with restraints (Physical, time related) to get a certain task done.
7. Help people upgrade their contact networks inherently helping them for future situations.
8. Provide a safe, monitored and continuously improving way for people to contact strangers for jobs.
9. Provide a rating system for users to be able to determine safer future business relations.
10. Produce income enough to maintain and service the software dependencies and updates.

IX. Synopsis

This domain came to be, because we believe it's an worthy opportunity to explore, the idea of making a platform for users to post their requests, and have them done by willing and capable participants. The domain of this project consists of a network of users within a small enough distance between them. The purpose of this project is to provide an on demand service for a particular task that a person might need to get done. This creates the opportunity for other people to earn income for their time and expertise. The application will provide the following functions to the user. The ability to post or accept jobs from the server. The ability to make a profile that will contain important info about the user : Name, Location, Expertise, Rating, etc. The ability to rate the the work done by one of the users. The ability to file a complaint about another user for its service. An more functions as they’re needed. There are two key events that happen within our application. The first one is a user requesting a job and the second one is a user accepting the job. From this event certain behaviour can occur. Let's call a user requesting a job behaviour R. Behaviour R send the job to the server (Behaviour S). All of this would be implemented in Scala

programming language, and REST APIs. For the end goal of providing user satisfaction, and generate revenue for our domain as well for any participants of our software.

X. Standard Compliance

A. . The standards that are going to be used in this domain are:

1. IEEE Standard for Software Maintenance.
2. IEEE Guide for Developing System Requirements Specifications.
3. IEEE Standard for Software Project Management Plan.

XI. Contracts

A. Payment rights

1. Methods of Payment

a) Credit Cards

(1) MasterCard

(2) Visa

(3) Others

b) PayPal

B. Advertisements Rights

1. Allowing ads of other Domains into ours.
2. Having ads of our domain in others.

C. User Agreement/Terms of service

1. A certain set of rules the user must follow.

D. Rights of Admission

1. We have the right who can, or cannot use our software.

E. Access Users info

1. credit cards
2. location
3. expertise
4. Income
5. Access to banking information.

XII. Design Brief

A. Users have to abide to our terms of service,failing to comply by actions such as fraud,or illegal activities, could result in permanent suspension and may involve legal action.

XIII. Log Book

A. 12 Feb 2020

- Initial meeting of team members.

B. 14 Feb 2020

- Set up some guide question, to fill out the proposal.

C. 17 Feb 2020

- Filled out each section of the document.

D. 18 Feb 2020

- Rechecked each part to see if everything is in order.

E. 27 Feb 2020

- Introduction for new team members and dividing workload.

F. Informative Requirements Development Documents.

I. Name, Place, Date

A. Name:

1. Bounty Board

B. Place:

1. University of Puerto Rico, Mayaguez District

C. Date:

1. Starting Date: February 12th, 2020
2. Estimated Goal Date: May 1st, 2020

II. Partners

A. Developers:

1. Developers capable of making distinct versions of our software:

a) Capable developers in charge of making the web application using REST and scala.js

b) Capable developers in charge of making a mobile version on both Android OS and Apple's IOS.
c) Capable developers in charge of maintenance of both the web versions and mobile versions.
d) Capable developers in charge of bug fixes and testing.

2. Our team consists of:
    A. Anthony Zapata
    B. Ebdiel Roman
    C. Jean Lugo
    D. Kevin Burgos
    E. Javier Maldonado
    F. Songiemar Garcia
    G. Carlos Machin
3. As the developers of this software we will follow our roles and meet those requirements to provide a broader availability, so that the end goal will be that more user would be able to use the software.

III. Current Situation

A. The current situation is that a large amount of the population is either too busy, too lazy or incapable of getting small tasks done and that these tasks could be accomplished by others who are more willing to do them and have the availability to do so provided, of course, they get compensated for their work.

IV. Needs and Ideas

A. Needs:

1. There is a part of the population that are in need of finding others to get any task they require done, therefore they will need a platform where they can post their requests and use the tools that the platform provides to facilitate their needs.

2. There is a part of the population that are in need of finding jobs because the need more income o are currently unemployed, therefore they will need a platform where they can accept request and use the tools that the platform provides to facilitate their needs.

B. Ideas:

1. The idea of this software is to present a user to user job/aid opportunity. Users that may require a certain task to be completed by someone will have the ability to post this task as a "Bounty Hunt" in exchange for monetary rewards. This software provides a profile based system for its users to identify other users that are willing to fulfill this task.
2. Create a web app that displays a board of tasks where users can then browse said tasks and decide if they would like to complete it. The app would also allow users to post their own tasks so that others can accept them.
3. Implementing this software so you can solve the need of needing a helping hand.
4. Implement a platform for the software that can be used on web browsers and mobile devices.

V. Concept and Facilities

A. Concepts - The following solution concepts were selected for the app which are the following

1. Provide the user the ability to post to the server.
2. Provide the user the ability to accept jobs from the server.
3. Present a user friendly GUI that will contain the jobs requested from a close enough location.

B. Facilities - It's essential to use the adequate software to complete our project. PostgreSQL will be used to handle the

database and server side of the application. For the front-end JavaScript and Angular framework were the language and framework of choice. To keep track of work schedule Scala, ClickUp and other software were chosen.

VI. Scope and Span

A. Scope:

1. The scope of our domain, is generally in software development, specifically a piece of software that can be used by the public. Therefore, l we will produce a piece of software that provides a platform for users to, place request for one time job, and to provide a safe and easy way to contract workers and to make monetary gain or service for the domain holders and the users.

B. Span:

1. The span of our domain, is to facilitate the users of our piece of software to place requests, removes request, accept request form others, place monetary incentive, check expertise of acceptor, for a task that needs to be done. In addition to those functions, you could check by GPSso you could find jobs near your point of origins. You can link your preferred method of payment of choice to provide an accessible way of money transfers between users. In addition, a rank system would be in place to see who are the most qualified for the job, and the requestor can choose between all of the applicants. All of this would be implemented using REST API, and Scala programming language. In order to provide a platform, for the users to find or create jobs from the necessity of the requestor.

VII. Assumptions and Dependencies

A. Assumptions:

1. It is assumed that our team will have enough manpower and know how of building and developing the web based software, its functions and their expected behaviours.
2. It is assumed that our team will have enough manpower and know how to meet the needs of maintenance.
3. It is assumed that our team will have enough manpower and know how to meet the expected requirements in:
    a) Performance
    b) Dependability
    c) Maintenance
    d) Platform
    e) Interface
    f) Documentation
4. It is assumed that our team will meet all of the requirements mentioned above to provide that the software works properly and efficiently.

B. Dependencies:

1. Both the quality and efficiency of the application is dependant on this as well as the speed and pace in which the application is completed from design to production.

VIII. Implicit/ Derivative Goal

A. Our goals:

1. Our goal consists of:
    a) To have a software that meet all of our basic requirements.
    b) To have a software that is quick, efficient such that it meets our set performance requirements.
    c) To have a software that require little maintenance and can operate for long periods of

time. Therefore, it must meet the set requirements in maintenance.

d) To have a software that is dependable, and meet that requirement.
e) To have a team that is constantly looking in way to better taylor the software based on the users demands.
f) To have a team that is determined to find ways to improve the software and add any additional feature whenever possible.
g) To be able to successfully send and receive information.
h) To be able to keep a backup of the information of the software and users information, in the case of any problems that may arise in the future.
i) To be able to make update at any time with relative ease.
j) To have met all of the requirement to provide platform where user can easily and safely post their task and others to accept them.

IX. Synopsis

A. This domain came to be, because we believe it's an worthy opportunity to explore, the idea of making a platform for users to post their requests, and have them done by willing and capable participants. The domain of this project consists of a network of users within a small enough distance between them. The purpose of this project is to provide an on demand service for a particular task that a person might need to get done. This creates the opportunity for other people to earn income for their time and expertise. The application will provide the following functions to the user. The ability to post or accept jobs from the

server. The ability to make a profile that will contain important info about the user : Name, Location, Expertise, Rating, etc. The ability to rate the the work done by one of the users. The ability to file a complaint about another user for its service. An more functions as they're needed. There are two key events that happen within our application. The first one is a user requesting a job and the second one is a user accepting the job. From this event certain behaviour can occur. Let's call a user requesting a job behaviour R. Behaviour R send the job to the server (Behaviour S). All of this would be implemented in Scala programming language, and REST APIs. For the end goal of providing user satisfaction, and generate revenue for our domain as well for any participants of our software.

X. Standard Compliance

A. . The standards that are going to be used in this domain are:

1. IEEE Standard for Software Maintenance.
2. IEEE Guide for Developing System Requirements Specifications.
3. IEEE Standard for Software Project Management Plan.

XI. Contracts

A. A contract is to be made with the developers and any additional personnel required in the development of the software be it for:

1. Maintenance personnel
2. Personnel in charge porting to other devices such as:
    a) Android
    b) IOS
3. Debuggers

B. A contract is to be made that ensures that the software works as expected and that the sensitive information of the user will be protected, for that contracting cyber security expert will prove beneficial to prevent data leaks and avoid compromising the users account thus avoiding potential legal issue.

XII. Design Brief

A. All of the personnel contracted will work on the software in different aspects of it:

1. A team of development personnel in charge of the main aspects(Variables, functions, behaviours, etc) and basic requirements.
2. A team of porting personnel in charge of bringing the software to a broader range of devices.
3. A team of maintenance personnel in charge of performing changes and/ or update needed to keep the software functional, and meeting the set requirements .
4. A team of optimization personnel in charge of meeting performance requirements.
5. A team of cyber security specialist to meet security requirements.

XIII. Log Book

A. 19 Feb 2020

1. Initial meeting of team members.

B. 20 Feb 2020

1. Set up some guide question
2. Agreeing on the direction the software should go.

C. 21 Feb 2020

1. Distributed work between team members.

D. 22 Feb 2020

1. Read chapter 1 & 2 as a refresher,
2. Determined the entities, events, functions, and behaviours.

E. 23 Feb 2020

1. Determined the requirements of the software
2. Determined the informative documents, descriptive documents, rough sketch, and table of contents

B. 1 Mar 2020

1. Completed Algebra portion

C. 4 Mar 2020

1. Completed Prediccates

D. 6 Mar 2020

1. Completed Axioms

F. Descriptive Rough Domain Sketches

1. App development is a concept where you try to solve a problem or fulfill a role or need using software.
2. Bounty Board is app with the end goal of posting requests and fulfilling them.
3. A post is a task uploaded to the platform.
4. Tasks can be anything that you are willing to pay to get done.
5. A profile is a summary of the person's information, contact info and qualification.
6. A contractor type profile is specifically for requesting tasks.
7. Acceptor is for applying for the task.
8. The contractor has final say on which applicant will be eligible to fulfill the task.
9. A profile has ranking based on the quality of the their past performances.
10. Rankings are based on the reviews left by the contactor.
11. Reviews are strict evaluations on the performance of a person's assigned task.
12. The the rank is proportional to the reviews and the amount of tasks done.
13. The reviewers are the parties involved in the strict evaluation of a task.
14. Reviews can be biased or unbiased.
15. Biased reviews are usually subjective to the person.
16. Unbiased reviews are strict and objective evaluation regardless of circumstances.

17. A review happens on both sides, on the performance of the acceptor's assigned task, and on the fairness/pay of the contractor.
18. A profile can have a ranking of contractor type or an acceptor type, independent from each other.
19. Apps can be developed as a web application and as a mobile application.
20. Apps can be runned on a variety of devices, such as mobile devices or within a web browser.
21. Apps can be plentiful, in particular there exist many apps like bounty Board, but our is unique because is based on a request, apply philosophy.
22. Services can be divided into specific categories.
23. Service Coins are the currency of Bounty Board.
24. Service Coins can be bought with real currencies.
25. The exchange rate is one to one with the respective currency in USD.
26. Jobs are paid in service coins.
27. Linking is a way to integrate a method of payment, that automatically converts money into service coins.

G. Concept Analysis of Rough Domain Sketch

1. An app developed platform is a place where users can submit a post and discuss with other users as well as reply to other users posts.
2. Bounty Board's platform is place where users submit request of tasks, and others apply for the task.
3. Bounty Board's community is a place for users in need of assistance and users to find jobs.
4. Bounty Board's community is not the place where you conduct the task.
5. A profile is a sensitive document that contains: full name, age, gender, location, cell phone number, email, social media integrations, brief description on what they specialize in, ranking and a resume.
6. The reviewer refers to the person who needs to evaluate another person, in this case it is contractor who reviews, they can review with

the following criteria, quality, efficiency, fairness, punctuality, responsibility, follows orders, skills, appearance and friendliness.

7. Acceptors reviews can evaluate in the following criteria: fairness, friendliness and pay.
8. A review is not an simple scale, rather it is relative to the written reviews. Any reviews with just numbers on a scale and no actual feedback are treated as spam and thus not considered for ranking.
9. Service coins are Bounty Board, currency, this to ensure that the money converted is legimite, and thus eligible for participating.
10. You can earn a random number of from 0-25 service coins by clicking ads on the screen once a day. This way we can fuel some of the traffic and economy in sustainable way.

H. Descriptive Domain Terminology

1. App
    A. Short for application and it is piece of software with a specific purpose, produced by developers to solve a problem.
2. Category
    A. Types of services.
3. Profile
    A. A profile is a sensitive document that contains: full name, age, gender, location, cell phone number, email, social media integrations, brief description on what they specialize in, ranking and a resume.
    B. There are two types contractors and acceptors.
4. Contractor
    A. A user that post task.
5. Acceptor
    A. A user that applies for task.
6. Ranking
    A. Ranking are a way for a users to determine if an applicant is reliable to fulfill a task. You can review in terms if they are a

contactors and acceptor, and their ranks are independent from each other.

7. Reviews
    A. To analyze and critique a aspect of a task. Be it biased or unbiased.
8. Report
    A. Report for breaching terms of services or misconduct.
9. Bounty Hunt
    A. A term whenever a task is accepted by both parties. Indicates that the fulfillment of the task is on the way.
10. Ads
    A. Short for advertisements, they are present in order to have servers running. They are meant to market other products in exchange for some extra revenue.
11. Service Coins
    A. Currency of Bounty Board, serves for payments where it is necessary. Transaction
    B. To exchange a currency between one user to another.
12. Exchange Rate
    A. The amount of worth of currency in terms of another. The exchange rate is one to one with the respective currency in USD.

I. Descriptive Domain Narrative
    1. The domain is an app developed platform which its end goal is to facilitate the users who need to a certain task done, however for whatever reason, you don't want to do it yourself. Aswell, if you are looking for jobs because you want to earn a little extra or are currently unemployed. We facilitate the hassle, by providing a platform of users willing to do work and others willing to give work. This is done easily by a system of posts(request) and applyments. The entities that we can observe are:

A. Rankings: which can be decomposed into a dependency of reviews. Rankings is part of a leaderboard system where the lower your number the higher you are and more likely to do jobs correctly.
B. User reviews: which can be decompose into the action of give another party a rating on how well they performed or how well are they treated.
C. Specifications are an objective to follow to make sure the task is completed and thus fulfilling both the users types roles.
D. Users are composed of information such as their profile,which contain their personal information and contact information.
E. Bounty Hunt is the main event in the software, here's where posting and accepting task happen, completing the task, reviews and monetary gain,

The functions described above are about posting tasks, applying for a task, and confirming the applicant for a Bounty Hunt. The events that follow would be that the applicant will complete the Task (Bounty Hunt) receive payment in an exchange of Service Coins from the contractor to the acceptor. Then, both engage in the review process, so that their ranking can be updated. Afterwards, the receiver can decide to the exchange shop to convert their Service Coins into their desired currency. At the end of this process, the objective is complete on both parties, a task is done and the acceptor is paid for their service.

J. Table of Contents

III. Algebra

Class MasterMapList

type

mapList, mapSublist, profileSublist, categorySublist, jobSublist, keys, user, moderator, profile category, jobs

values

mapList= {mapSublist}

mapSubList= {profileSublist, categorySublist, jobSublist}

keys= by{profile, category, job}

userSet(key, arg, user): user X mapList X mapSubList X userSublist X profile ↠ userSubList

userGet(key, arg, user): user X mapList X mapSubList X userSublist↠ profileSublist

userPut(key, arg, user): user X mapList X mapSubList X user➡user SubList

userRemove(key, arg, user): user ∨ moderator X mapList X mapSubList X user ↬ userSubList

categorySet(arg): moderator X mapList X mapSubList X category ↠categorySublist

categoryGet(arg): user X mapList X mapSubList ↠ categorySubList

categoryPut(key, arg, category): moderator X mapList X categorySubList X category➡categorySublist

categoryRemove(key, arg, category): moderator X mapList X categorySubList X category ↬ categorySubList

jobSet(arg): user ∨ moderator X mapList X mapSubList X categorySubList X job ↠jobSublist

jobGet(arg): user X mapList X mapSubList X categorySubList ↠ jobSubList

jobPut(key, arg, category, job): user X mapList X categorySubList X jobSubList X job ➡jobSublist

jobRemove(key, arg, category, job): user ∨ moderator X mapList X categorySubList X jobSublist X job ↬ jobSubList

sortby(Data, args): user X mapList X mapSublist X categorySublist ⋁ jobSublist ⋁ userSublist ➡ sortedSublist

isEmpty: Maplist ↠ **Bool**

Maplist.remove ➡ **Chaos**

Maplist.isEmpty.remove ➡**Undefined**

Axioms:

∀Maplist.isEmpty ↠ **Bool**

∀Maplist.remove ➡ **Chaos**

∀Maplist.isEmpty.remove ➡**Undefined**

end

The type of algebra used in this domain is that of a Map & List, this structure is composed of elements that are Maplists. Theres is a particular hierarchy in this structure, for example: the master maplist that contains a user sublist that contains profile data. The master maplist also contains the category map sublist which in turn contains another mapsublist as it's elements, which is the job Sublist. The user map sublist contains reference to all of the information provided when creating an account, it contains profile objects. The category map sublist contains a nested maplist, containing the jobs. The job map sublist contains, job objects, which in turn contains any specifications, pay and requirements. For the general domain functions, the get(key, args, data type) functions, simply returns the corresponding map Sublist or object type. For example category map subList returns a job subList and job subList returns job objects. For the set(key, args, data type) functions, it modifies any existent element and replace it with the data given by the arguments. For the add(key, args, data type) functions, it add a new entry to the existent desired location of the hierarchy, for example: adding a job would place it on the job mapSubList and adding a user would place it on the user mapSublist. For the remove (key, args, data type) function, it remove any specific element and replace the empty spot with the nearest ordered element. For the case that an elements has a subelements, operations would requires work from the lowest ranking part in the hierarchy to prevent data loss.

The axioms here in particular are to state what is the expected result of the different types of operation performed on the different mapList.

IV. Predicates

class: User

type:

user, info, profile, mapList, mapSublist, userSublist

values:

inputFirstName: user X display ➡ textbox

inputLastName: user X display ➡ textbox

inputEmail: user X display ➡ textbox

inputPhoneNumber: user X display ➡ textbox

inputPassword: user X display ➡ textbox

inputLocation: user X display ➡ textbox && “Input Zip Code”

linkSocialMedia: user X display ➡ “Connect to:” (Facebook || Twitter || Instagram || LinkedIn)

inputDateOfBirth: user X display ➡ textbox(day), textbox(month), textbox(year)

inputDescription: user X display ➡ textbox

uploadResume: user X select_file

selectGender: user X display ➡ dropdown(Male || Female || Prefer Not to Specify)

ShowProfile:

{user} ε info

Axioms:

∀mapList= ∃ mapSublist

∀ mapSublist= ∃ userSublist

∀ userSublist= ∀ ∃! user

∀user.info= ∃ FirstName(x) ∧ ∃ LastName(x) ∧ ∃ age(x) ∧ ∃ gender(x)

∃! Email(x) ∧ ∃ PhoneNumber(x) ∧ ∃ Password(x) ∧ ∃! Location(x) ∧ ∃! SocialMedia ∧ ∃ Description ∧ ∃! Resume(x)

end

In this “User” Class, there are different functions in which the user can input the required information and upload their resume. This class is utilized when the person initially signs up for the Bounty Board, to create a profile.

The axiom here in particular denotes that it should follow a certain mapList hierarchy. A maplist should contain at least one sublist, and one of those sublist should be a user sublist which in turn must contains a multitude of unique users. The user has a particular information, and every user should have a first name, last name, age, gender, a unique email, phone number, password, a unique location of origin, at least one unique social media profile integration, a self description and a unique resume. Which we can conclude that every user should contain one unique profile.

==========

class: Profile

type: user, profile, display, ranking

values:

getFirstName: user X display ➡ firstName

getLastName: user X display ➡lastName

getContactInfo: user X display ➡PhoneNumber

getLocation: user X display ➡locationProximity

getSocialMedia: user X display ➡ socialMediaLinks

getDescription: user X display ➡Description

getRanking: user X display ➡Ranking

viewResume: user X display ➡Resume

Axioms:

$\forall$user(x)..info.profile= $\exists !$ profile(x)

($\neg \exists$ profile(x) ➡ $\neg \exists$ user.info.profile) $\therefore \neg \exists$ user(x)

$\forall$user.info.profile= $\exists$ FirstName(x) $\wedge$ $\exists$ LastName(x) $\wedge$ $\exists$ PhoneNumber(x) $\wedge$ $\exists !$ Location(x) $\wedge$ $\exists !$ SocialMedia $\wedge$ $\exists$ Description $\wedge$ $\exists !$ Ranking $\wedge$ $\exists !$ Resume(x)

end

The “Profile” Class takes the information inputted in the “User” class, while limiting some of the information available. This is how other Users would be able to view information of whom they’re taking a Request from, or vice versa: a User with a Request would be able to view information of potential candidates, in order to select one for their Request.

The axioms for this part describes if an user has the information given, then there must exist a public profile. If the profile does not exist then user information does not exist as

well, therefore user must should not exist as well. The user profile show the relation of the information given by the user and what is show on the profile, so it should mean a first name must exist, last name must exist, phone number, unique location, unique social media integration, a personal bio-description, resume, and current overall ranking.

==========

class: Ranking

type: user, ranking, display, profile

values:

addReview: user X display ➡review page options

inputComment: user X display ➡textbox

inputScore: user X display ➡(0 - 10 stars)

calcNewRanking: profile*getRanking({avg_(userScore, inputScore)})

axioms:

∀user(x)= ∃! ranking

∃ inputScore= ∃ Review

∃ Review= ∃ ranking

¬∃ Review= ¬∃ ranking

∀ Ranking= ¬input Comment

end

Once a Request is completed, the "Ranking" Class is going to give both parties the opportunity to review one another other in a numeric system (0-10stars) and, optionally, leave a comment. Afterwards, the review will be processed and the score will be calculated, giving each user's a new Ranking.

The axioms here are for describing the relationship between reviews and ranking. It is expressed here that if there exist a review than there exist ranking, but the inverse is true as well, therefore is a review does not exist att all for the user then that user has no ranking yet. Ranking are bounded by the constraints of Reviews, not by comments.

==========

class: BountyHunt

type:

profile, user, display, ranking, request, job, category, mapList, mapSublist, categorySublist, JobSublist, ranks 1-11, applicants, applicantsSublist.

values:

sortBy(Ranking): user X display X mapList X userSublist➡ by_profileRanking * request

sortBy(category): user X display X mapList X categorySublist ➡ by_category* request

sortBy(Pay): user X display X mapList ➡ by_pay * request

sortBy(Applicant): user X display X mapList ➡ by_ranking * request(applicants)

placeRequest: user X display X mapList ➡ place_request

claimRequest: user X request X display ➡ “Application submitted.”

confirmApplicant: user X request X display ➡ “Selected!”

completedRequest: user X request X display ➡ “Completed!” // (reviewRequester || reviewApplicant)

reviewRequester: user X display ➡ Ranking(Requester)

reviewApplicant: user X display ➡ Ranking(Applicant)

selectPay: user X display ➡ “Select Currency:” {payServiceCoins || payRealMoney}

payServiceCoins: user X input_methodOfPaymemt(amount)

payRealMoney: user X input_methodOfPaymemt(amount)

Axioms:

∀user.profile= ∃ Ranking(x)

∀user in applicantsSublist *display

invoke sortby(Type, Data)➡

if(Data.equals(“0 Star”)

result: applicantSublist ∩ rank11

if(Data.equals(“1 Star”)

result: applicantSublist ∩ rank10

if(Data.equals(“2 Star”)

result: applicantSublist ∩ rank9

if(Data.equals(“3 Star”)

result: applicantSublist ∩ rank8

if(Data.equals(“4 Star”)

result: applicantSublist ∩ rank7

if(Data.equals(“5 Star”)

result: applicantSublist ∩ rank6

if(Data.equals(“6 Star”)

result: applicantSublist ∩ rank5

if(Data.equals(“7 Star”)

result: applicantSublist ∩ rank4

if(Data.equals(“8 Star”)

result: applicantSublist ∩ rank3

if(Data.equals(“9 Star”)

result: applicantSublist ∩ rank2

if(Data.equals(“10 Star”)

result: applicantSublist ∩ rank1

∀job.category=∃Category(x)

∀jobs in CategorySublist *display

invoke sortby(Type, Data)➡

if(Data.equals(User Input)

∃UserInput➡

result: jobSublist ∩ User Input

∀Request= ∃! Job(x)

∀Request= ∃ Description(x)

∀Job= ∃! Description(x)

∀CompletedRequest= ∃! Review

∀Completed Request= ∃payment(Type)

∀Review= ∃Updated Rank

end

This " BountyHunt" Class the act of requesting a job and accepting a job occurs. In this page you can sort for what are you looking for. You can apply for the jub and if the applicant is accepted then you can complete the task. After that the review stage and the payment stage can happen.

The axioms for this class describe the ability to have multiple types of sorting options such as sorting by rank, and sort by category of job. All users have a separate ranking, which then can be used to identify the users in a rank sorted search. Users can also sort based on category of job which will return a subset of jobs with the specific properties required by the user. All requests must come with a description, all jobs provide a unique description. Once a job is completed it will receive a review specific to that job. All jobs must be paid if they are completed, and all reviews will affect in some manner the rank of the supplier user. Rankings go from 0 star to 10 star ranks.

==========

class: Request

type: jobs, profile, user, display

values:

viewProfile: user X display ➡ profile

getDescriptionR: user X display ➡ inputedText(Request description)

redirectToBoard: user X display ➡ jobs

Axioms:

∀Request= ∃! Job(x)

∀Request= ∃ Description(x)

∀Job= ∃! Description(x)

end

In this "Request" Class, it gathers information from the User placing a Request and

displays the description they have inputted of the task: making it visible to those in the Bounty Board.

The axiom for this particular class defines the relationship between Request and jobs. For all requests posted by an arbitrary user then, the exist one unique job with a unique job description. To post a request there must exist a description, if these boundaries are exceeded or requirements are not met then a job should not exist.

==========

class: RankSort

type:

user, display, mapList, mapSublist, jobSublist, userSublist, Ranking, applicantSublist, rank1, rank2, rank3,rank4,rank5,rank6,rank7,rank8,rank9,rank10,rank11

values:

sortBy(Ranking): user X display X mapList X userSublist➡ by_profileRanking * request

Axioms:

∀user.profile= ∃ Ranking(x)

∀user in applicantsSublist *display

invoke sortby(Type, Data)➡

if(Data.equals(“0 Star”)

result: applicantSublist ∩ rank11

if(Data.equals(“1 Star”)

result: applicantSublist ∩ rank10

if(Data.equals(“2 Star”)

result: applicantSublist ∩ rank9

if(Data.equals(“3 Star”)

result: applicantSublist ∩ rank8

if(Data.equals(“4 Star”)

result: applicantSublist ∩ rank7

if(Data.equals(“5 Star”)

result: applicantSublist ∩ rank6

if(Data.equals(“6 Star”)

result: applicantSublist ∩ rank5

if(Data.equals(“7 Star”)

result: applicantSublist ∩ rank4

if(Data.equals(“8 Star”)

result: applicantSublist ∩ rank3

if(Data.equals(“9 Star”)

result: applicantSublist ∩ rank2

if(Data.equals(“10 Star”)

result: applicantSublist ∩ rank1

end

This Class “RankSort” check all the applicants of a request and sort it from highest Rank(Rank1) or lowest (Rank11) or anything in between.

The axioms for this class is that when choosing a sorting type, in this case Rank, you can choose what rank you are looking for and makes an new sublist based on the selected rank.

==========

class: CategorySort

type:

user, job, display, mapList, mapSublist, jobSublist, userSublist, CategorySublist, Ranking

values:

sortBy(category): user X display X mapList X categorySublist ➡ by_category* request

Axioms:

∀job.category= ∃Category(x)

∀jobs in CategorySublist *display

invoke sortby(Type, Data)➡

if(Data.equals(User Input)

∃UserInput➡

result: jobSublist ∩ User Input

end

This class “CategorySort” sorts by finding job in given category by the user. Then placing them on its own sublist.

The axioms for this class is that when choosing a sorting type, in this case Category, you can choose what category you are looking for and makes an new job sublist based on the selected category, but first it checks if the User input exists.