–––––a

**ADDIS ABABA UNIVERSITY**

**COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE**

**SCHOOL OF INFORMATION SCIENCE**

**DEPARTMENT OF INFORMATION SYSTEM**

**Title:**  Local Freelance System

Submitted in partial fulfillment of Bachelor’s Degree in Information Systems (Industrial Project-2)

**Group members:**

Nabil Mohammed……….…...NSR/2848/10

Mulugeta Adamu …………....NSR /7036/10

Yohans Degu……………..….NSR/3893/10

Yigermal Bizuneh……...........NSR/9152/10

Yohannes Hunegnaw…..……NSR/1385/10

Daniel Lakew………………..NSR/1998/10

**Addis Ababa University**

**School of Information Science**

**Industrial project Progress report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | |  | | |
| **Title** | |  | | |
| **Advisor** | | **Name** | | **Sign** |
|  | |  |
| **Project team members list** | | | | |
|  | **Name** | | | **Sign** |
| **1** |  | | |  |
| **2** |  | | |  |
| **3** |  | | |  |
| **4** |  | | |  |
| **5** |  | | |  |
| **6** |  | | |  |
|  | | **Chapter Number** | **Major topics** | |
| **Chapters submitted** | |  |  | |
|  |  | |
|  |  | |
| **Chapters revised** | |  |  | |
|  |  | |
|  |  | |
| **Major works remaining** | |  |  | |
|  |  | |
|  |  | |

**Examination Board**

**Advisor Name:**

**Signature:**

**Date:**

**Examiner Name:**

**Signature:**

**Date:**

**Examiner Name:**

**Signature:**

**Date:**

**Acknowledgement**

Primarily, praise and gratitude to God, the Almighty, for his abundant gifts that enabled us to complete our half-semester project successfully. We would like to convey our heartfelt gratitude to our adviser, Lemlem Hagos, who has dedicated herself entirely to guiding the team to success. Throughout the writing of the paper, we are grateful for her unwavering patience, encouragement, and unreserved and important advice. She supported us through all of the challenges we had, from developing the project proposal until the completion of this phase of the project, with keen intellect and understanding.

Finally, and most significantly, we want to express our gratitude to our families and friends for their unwavering support and encouragement throughout our studies. Finally, let us express our gratitude to those persons who directly or indirectly assisted us in completing this project within the time constraints.

**Table Contents**

[**List of tables** iii](#_Toc83081594)

[**List of figures** iii](#_Toc83081595)

[**5.2 Introduction to the Chapter** 4](#_Toc83081596)

[**5.3.2 View** 7](#_Toc83081597)

[**5.3.3 Controller** 8](#_Toc83081598)

[**5.4. Design Class Modeling** 8](#_Toc83081599)

[**5.4.1** **Class diagram** 8](#_Toc83081600)

[**5.4.2** **Class description** 10](#_Toc83081601)

[**5.5. Collaboration Diagram** 15](#_Toc83081602)

[**5.6 Component diagram** 18](#_Toc83081603)

[**5.7 Deployment Diagram** 19](#_Toc83081604)

[**5.8. Relational Persistent Model** 20](#_Toc83081605)

[**5.9. User Interface** 22](#_Toc83081606)

[**5.9.1. User Interface Flow Diagram** 22](#_Toc83081607)

[**5.9.2. User Interface Design** 23](#_Toc83081608)

[**CHAPTER 6** 28](#_Toc83081609)

[**6. OBJECT ORIENTED IMPLEMENTATION** 28](#_Toc83081610)

[**6.1 Introduction** 28](#_Toc83081611)

[**6.2 Implementation Technology** 28](#_Toc83081612)

[**6.4 Testing and testing procedure** 28](#_Toc83081613)

[6.4.1 Unit Testing 28](#_Toc83081614)

[**6.4.2 Integration Testing** 29](#_Toc83081615)

[**6.4.3 System Testing** 29](#_Toc83081616)

[**6.5 Deployment/Installation process** 32](#_Toc83081617)

[**6.5.1 Hardware Acquisition** 32](#_Toc83081618)

[**6.5.2 Software Acquisition** 32](#_Toc83081619)

[**CHAPTER 7** 33](#_Toc83081620)

[7. **Conclusion and Recommendations** 33](#_Toc83081621)

[**7.1. Conclusion** 33](#_Toc83081622)

[**7.2. Recommendation** 33](#_Toc83081623)

[**Reference** 34](#_Toc83081626)

[**Appendex** 34](#_Toc83081627)

# **List of tables**

[Table 1 : sign up test case 28](#_Toc82686015)

[Table 2: login test case 29](#_Toc82686016)

[Table 3: announce project test case 30](#_Toc82686017)

# **List of figures**

[Figure 1:MVC class type Architecture: 6](#_Toc83081640)

[Figure 2:class diagram 10](#_Toc83081641)

[Figure 3:Announce project collaboration diagram 16](#_Toc83081642)

[Figure 4:Deliver project by service provider collaboration diagram 16](#_Toc83081643)

[Figure 5:Issue ticket by service provider collaboration diagram 16](#_Toc83081644)

[Figure 6:message by service provider collaboration diagram 17](#_Toc83081645)

[Figure 7:Raise dispute collaboration diagram 17](#_Toc83081646)

[Figure 8:componet diagram 18](#_Toc83081647)

[Figure 9:Deployment Diagram 19](#_Toc83081648)

[Figure 10:ER diagram 21](#_Toc83081649)

[Figure 11:user Interface flow diagram 22](#_Toc83081650)

[Figure 12:Landing page 23](#_Toc83081651)

[Figure 13:Login page 24](#_Toc83081652)

[Figure 14:Service provider sign up page 25](#_Toc83081653)

[Figure 15:Service seeker sign up page 26](#_Toc83081654)

[Figure 16:Home page for service provider 26](#_Toc83081655)

[Figure 17:home page service seeker 27](#_Toc83081656)

[Figure 18:Home page for Admin 27](#_Toc83081657)

**CHAPTER 5**

5. **Object Oriented Design**

**5.1 Review of the First Phase**

We have worked on four major chapters in the first phase of the industrial project: Project proposal, Business Area Analysis and Requirement Definition, and Object-Oriented Analysis. The first chapter includes an introduction and an objective. The overall goal of this project is to create a web-based platform for service exchange between Ethiopian service providers and service seekers, as well as to increase job creation.

In the second chapter, we identified the organization's activities, business processes, existing system problems, forms and reports used, and the organization's players. Following that, we determined the system's functional needs in order to tackle the difficulties revealed throughout the analysis process with the existing system. The non-functional requirements of the system, as well as collaboration among distinct classes, are modeled in this chapter.

Following the organization's business area analysis, the third chapter presented the object-oriented system development and design phase of the local freelance system. This chapter covered some concepts of object-oriented design as an introduction and System use case along with its diagram and description, the identification of actors that can use the system, identification of user interface for the system, the conceptual modeling which includes a class diagram and the description of the class diagram and sequence diagram of the systems use case as well as the user interface prototyping of the new system to be developed.

## **5.2 Introduction to the Chapter**

In this chapter, we covered the major Object-Oriented Design artifacts, such as class type architecture, class modeling, collaboration modeling, component modeling, deployment modeling, user interface design, and other design artifacts, in order to fill the gap between analysis and implementation.

The goal of Object-Oriented Design (OOD) is to figure out how to build a system and gather the data needed to create the actual implementation. By emphasizing on how the system will be constructed, this differs from analysis, which focuses on knowing what will be built.

**5.3 Class type Architecture**

We have used M-V-C architecture (Model View Controller) because of MVC is used to decompose the whole system into three subsystems (model) these are Model View and Controllers.

MVC patterns separate the input, processing, and output of an application. This model divided into three interconnected parts called the model, the view, and the controller. All of the three above given components are built to handle some specific development aspects of any web or .net application development.

In the MVC application development, the controller receives all requests for the application and then instructs the model to prepare any information required by the view. The view uses that data prepared by the controller to bring the final output

We have a variety of reasons to adopt MVC as our development architecture some of them are the following.

**Faster development process:** since MVC supports rapid and parallel development it is possible that one programmer can work on view while the other can work on controller to create the business logic of web application. Hence this way the applications developed using the MVC model can be completed faster than applications that are developed using other development patterns.

**Ability to Provide Multiple Views:** In the MVC Model, you can create multiple views for a model. Today, there is an increasing demand for new ways to access your application and for that, MVC development is certainly a great solution. Moreover, in this method, Code duplication is very limited because it separates data and business logic from the display.

**The Modification Does Not Affect the Entire Model:** For any web application; the user interface tends to change more frequently. Moreover, adding a new type of view are very easy in the MVC pattern because the Model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture

.

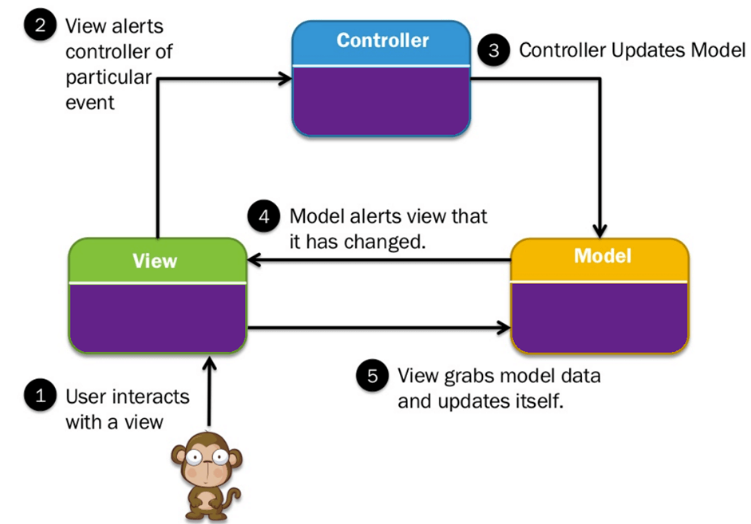


Figure 1:MVC class type Architecture:

**5.3.1 Model**

This level of MVC responsible for maintaining data. The model is actually connected to the database so anything you do with data. Adding or retrieving data is done in the model component. It responds to the controller requests because the controller never talks to the database by itself. The model talks to the database back and forth and then it gives the needed data to the controller.

**List of models**

* Admin
* Bid
* Dispute
* Faq
* Message
* Notification
* Policy
* Payment
* Project
* Rate
* Service provider
* Service seeker
* Ticket
* Transfer request
* User

### **5.3.2 View**

View actually generates UI or user interface for the user. So at web applications when we think of the view component just think the Html/CSS part. Views are created by the data, which is collected by the model component, but these data are not taken directly but through the controller, so the view only speaks to the controller.

**Lists of Views**

* Landing page
* Login page
* FAQ page
* Policy Page
* Announce Project Page
* Update Profile page
* View project page
* View dispute Page
* View support ticket page
* View service provider page
* Create Admin Page
* View bids page
* View message page
* View notification page
* Process transactions page
* Submit project page

### **5.3.3 Controller**

The controller is a level that acts as the brain of the entire MVC system. A controller also acts as a link between a user and the system. It provides the user with the input by providing appropriate views to present it appropriately on the screen. The controller understands user output, converts it into the appropriate messages and passes the same to views.

**List of Controllers**

* Main Controller
* Admin Controller
* Service Seeker Controller
* Service Provider Controller

## **5.4. Design Class Modeling**

Class model shows the classes of the system, their interrelationships (including, inheritance, aggregation and association), and the operation and attributes of the classes. (Mayer, 1997)

A class is a representation of an object. To describe a class, we define its attributes and methods. Attributes are the information stored about an object while methods are what the object or the class does. The purpose of design-class modeling is to model static structure of the software that will be built. The only difference with the analysis version of class modeling is the focus on the solution instead of on the problem domain.

### **Class diagram**

Class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (methods), and the relationships among objects [1].

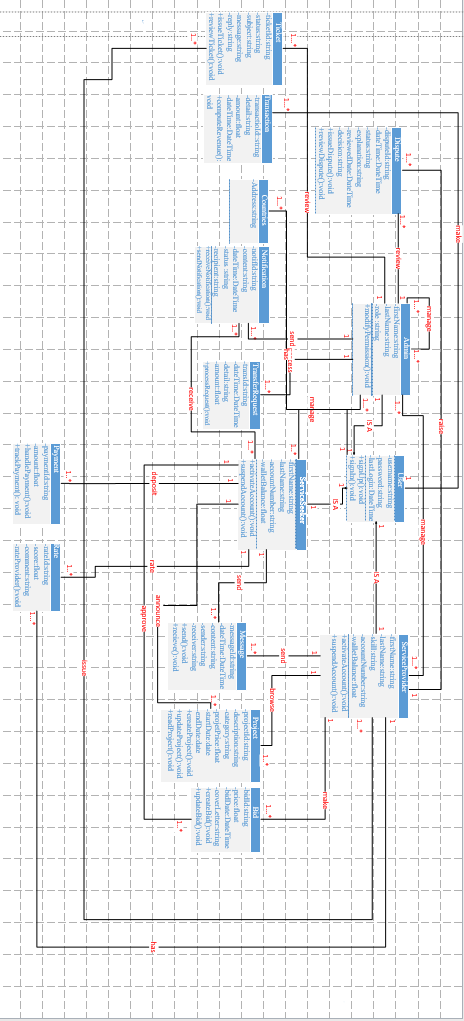


Figure 2:class diagram

### **Class description**

In this section, we will describe the classes depicted in the class diagram.

**Class name: User**

It is the class which is responsible to create user account, and sign in.

**Attributes**

* **username:** Identifies a name which is used for authentication purpose.
* **password:** Identifies a combination of characters which is used for authentication purpose.
* **lastLogin**: represent the last time the user has logged in.

**Methods**

* **signUp ():** creates account for the user.
* **signIn ():** the user login to the system.

**Class name: Admin**

It is a class which is responsible for managing the system.

**Attributes**

* **firstName**: Identifies the first name of the admin.
* **lastName:** Identifies the last name of the admin.
* **role:** Identifies the role of the admin.

**Methods**

* **modifyPermission():** main admin or authorized admin can modify the permission of other admin users.

**Class name: ServiceSeeker**

It is the class which is responsible to represent service seeker.

**Attributes**

* **firstName:** Describe the first name of service seeker.
* **lastName:** Describe the last name of service seeker.
* **accountNumber:** Identify bank account number of service seeker.
* **walletBalance:** Identify the wallet balance of the service seeker.

**Methods**

* **activateAccount ():** resumes the account.
* **suspendAccount ():** disables the account.

**Class name: ServiceProvider**

It is the class, which is responsible to represent service provider.

**Attributes**

* **firstName:** Describe the first name of service provider.
* **lastName:** Describe the last name of service provider.
* **skill:** Identifies the skills of service provider.
* **accountNumber:** Identify bank account number of service provider.
* **walletBalance**: Identify the wallet balance of the service provider.

**Methods**

* **activateAccount ():** resumes the account.
* **suspendAccount ():** disables the account.

**Class name: Project**

It is a class, which is responsible to represent a project.

**Attributes**

* **projectId:** identify specific project.
* **description:** refers to the project description
* **category:** refers to the category or area of the project.
* **projectPrice:** refers to the final agreed price to accomplish the projects.
* **startDate:** refers to the date in which the project started.
* **endDate:** refers to the date in which the project finished.

**Methods**

* **createProject():** enables to announce or offer new project.
* **updateProject():** update the status of the project.
* **readProject():** retrieves the details of the project.

**Class name: Bid**

It is a class which is responsible to represent a bid.

**Attributes**

* **bidId:** identify specific bid.
* **bidDate**: date and time in which the bid is made.
* **price:** refers to the price at which the service provider is willing to offer a project.
* **coverLetter:** refers to the description of the bid.

**Methods**

* **createBid():** allows to bid on projects.
* **updateBid():** update the status of the bid.

**Class name: Dispute**

It is a class, which is responsible to represent a dispute.

**Attributes**

* **disputeId:** refers to id of the dispute.
* **dateTime:** refers to the date and time in which the dispute is raised.
* **status:** refers to the status of a dispute, whether reviewed or not.
* **explanation:** refers to the description of the dispute.
* **reviewedDate:** refers to the date and time in which the dispute reviewed.
* **decision**: refers to the decision made by the admin for resolving the dispute.

**Methods**

* **issueDispute():** allows to raise a new dispute.
* **reviewDispute():** allows to resolve a dispute.

**Class name: Ticket**

It is a class, which is responsible to represent a ticket.

**Attributes**

* **ticketId:** refers to the id of the ticket.
* **status:** refers to the status of ticket whether reviewed or not.
* **subject:** refers the topic of the ticket.
* **message:** refers to the content of ticket.
* **reply:** refers to the reply content of ticket.

**Methods**

* **issueTicket():** allows to raise a new ticket.
* **reviewTicket():** allows to resolve a ticket.

**Class name: TransferRequest**

It is a class, which is responsible to represent a fund transfer request.

**Attributes**

* **transId:** refers to the id of fund transfer request.
* **dateTime:** refers to the date and time in which the request is sent.
* **detail:** refers to the detail of the request.
* **amount:** refers to the amount of fund to be transferred.

**Methods**

* **processRequest():** allows to update the status of a transferRequest.

**Class name: Notification**

It is a class, which is responsible to represent a notification.

**Attributes**

* **notifId:** refers to notification id.
* **content:** refers to the content of notification.
* **dateTime:** refers to date and time in which notification is created.
* **status:** refers to the status of notification whether it is read or not.
* **recipient:** refers to the notification recipient.

**Methods**

* **receiveNotification():** reads notification content.
* **sendNotification():** creates a new notification.

**Class name: Message**

It is a class, which is responsible to represent a message.

**Attributes**

* **messageId:** refers to the id of the message.
* **content:** refers to the content of the message.
* **sender:** refers to the sender of the message.
* **receiver:** refers to the receiver of the message.
* **dateTime:** refers to date in which message.

**Methods**

* **receive():** reads message data.
* **send():** creates a new message.

**Class name: Faq**

It is a class, which is responsible to represent a frequently asked question.

**Attributes**

* **faqId:** refers to the id of frequently asked question.
* **question:** refers to the content of the question.
* **answer:** refers to the answer of the question.
* **category:** refers to the category of the question.

**Methods**

* **editFaq():** updates Faq.

**Class name: Policy**

It is a class, which is responsible to represent a policy.

**Attributes**

* **policyId:** refers to the id of the policy.
* **name:** refers to the name of the policy formulated.
* **file:** refers to the file that contains the statements of the policy.

**Methods**

* **editPolicy():** updates policy.

## **5.5. Collaboration Diagram**

The collaboration diagram is used to show the relationship between the objects in a system [1]. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming [1]. An object consists of several features. Multiple objects present in the system are connected to each other [1]. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system [1].

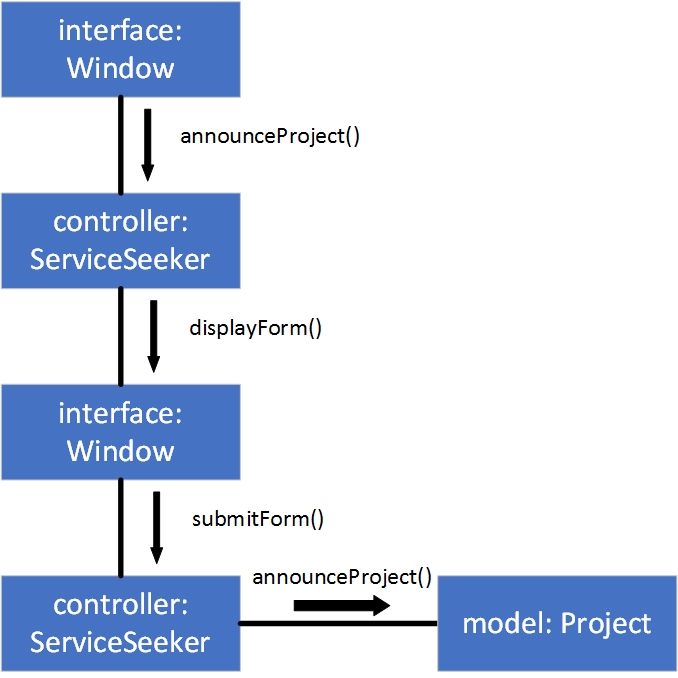
1. ****Announce project collaboration diagram

Figure 3:Announce project collaboration diagram

1. Deliver project by service provider collaboration diagram

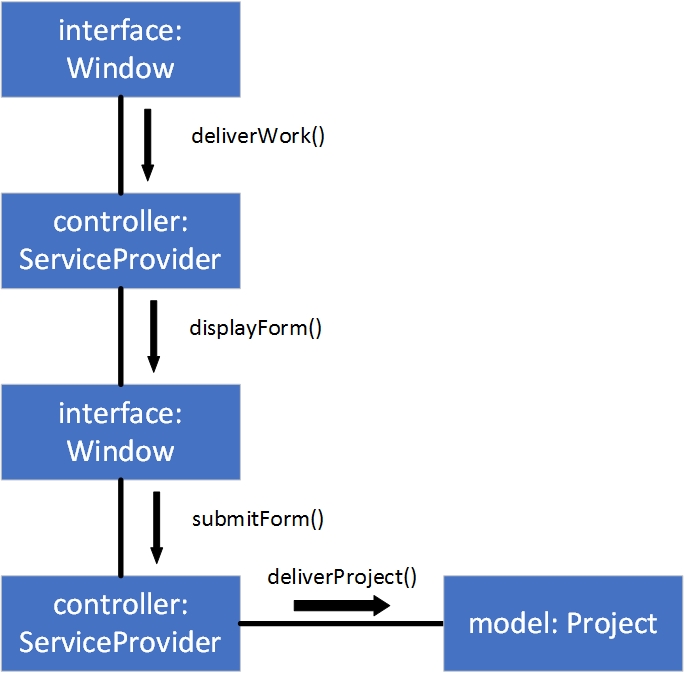


Figure 4:Deliver project by service provider collaboration diagram

1. Issue ticket by service provider collaboration diagram

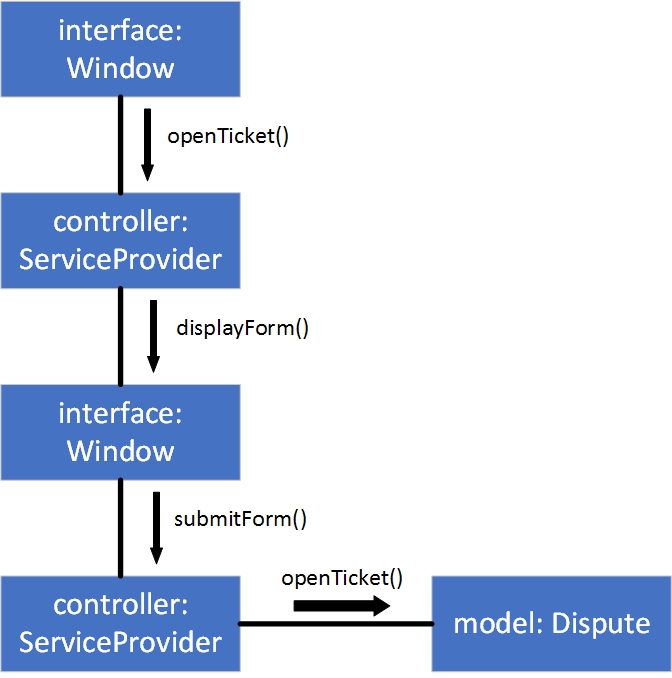


Figure 5:Issue ticket by service provider collaboration diagram

1. Send message by service provider collaboration diagram

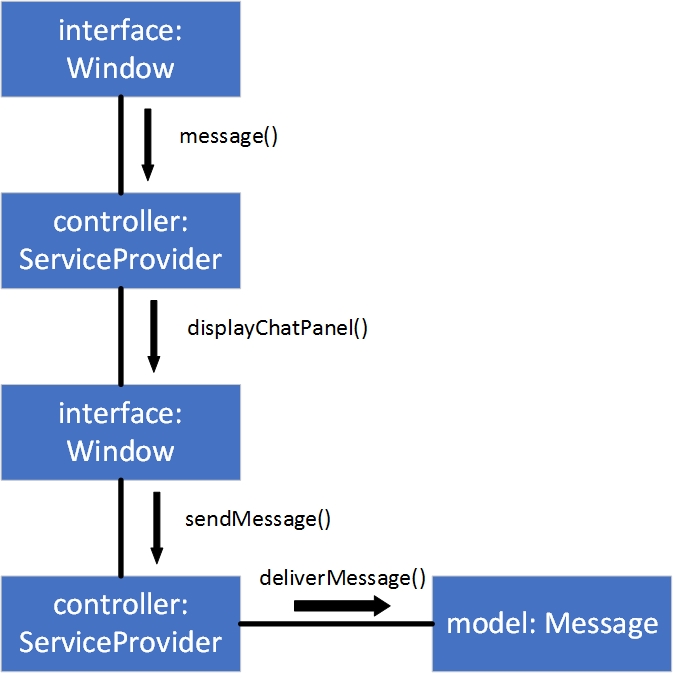


Figure 6:message by service provider collaboration diagram

1. Raise dispute collaboration diagram

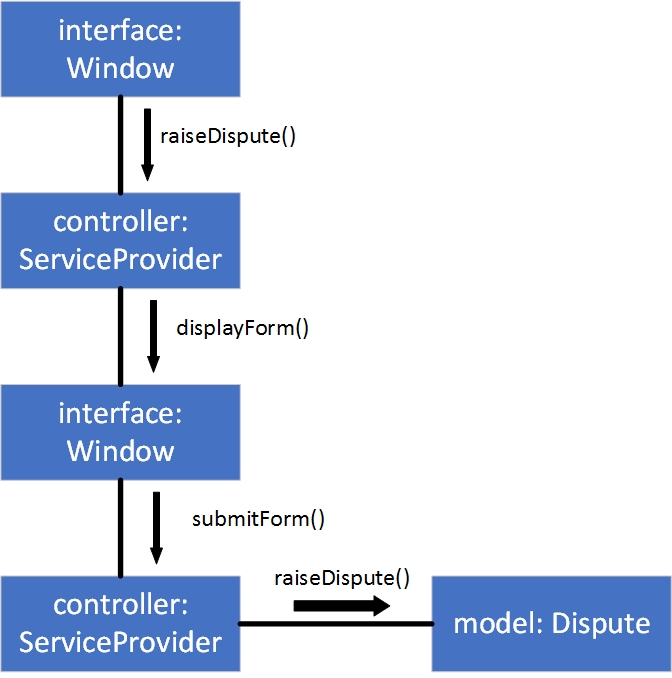


Figure 7:Raise dispute collaboration diagram

## **5.6 Component diagram**

**A component diagram** displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors.

**A component** is a replaceable and executable piece of a system whose implementation details are hidden. A component provides the set of interfaces that a component realizes or implements.

The interface is a named set of public features. It separates the specification of functionality from its implementation by a class diagram or a subsystem. Interfaces either can be provided or required interfaces

**Component diagrams** are used for,

* To represent the components of any system at runtime.
* It helps during testing of a system.
* It visualizes the connection between various components.

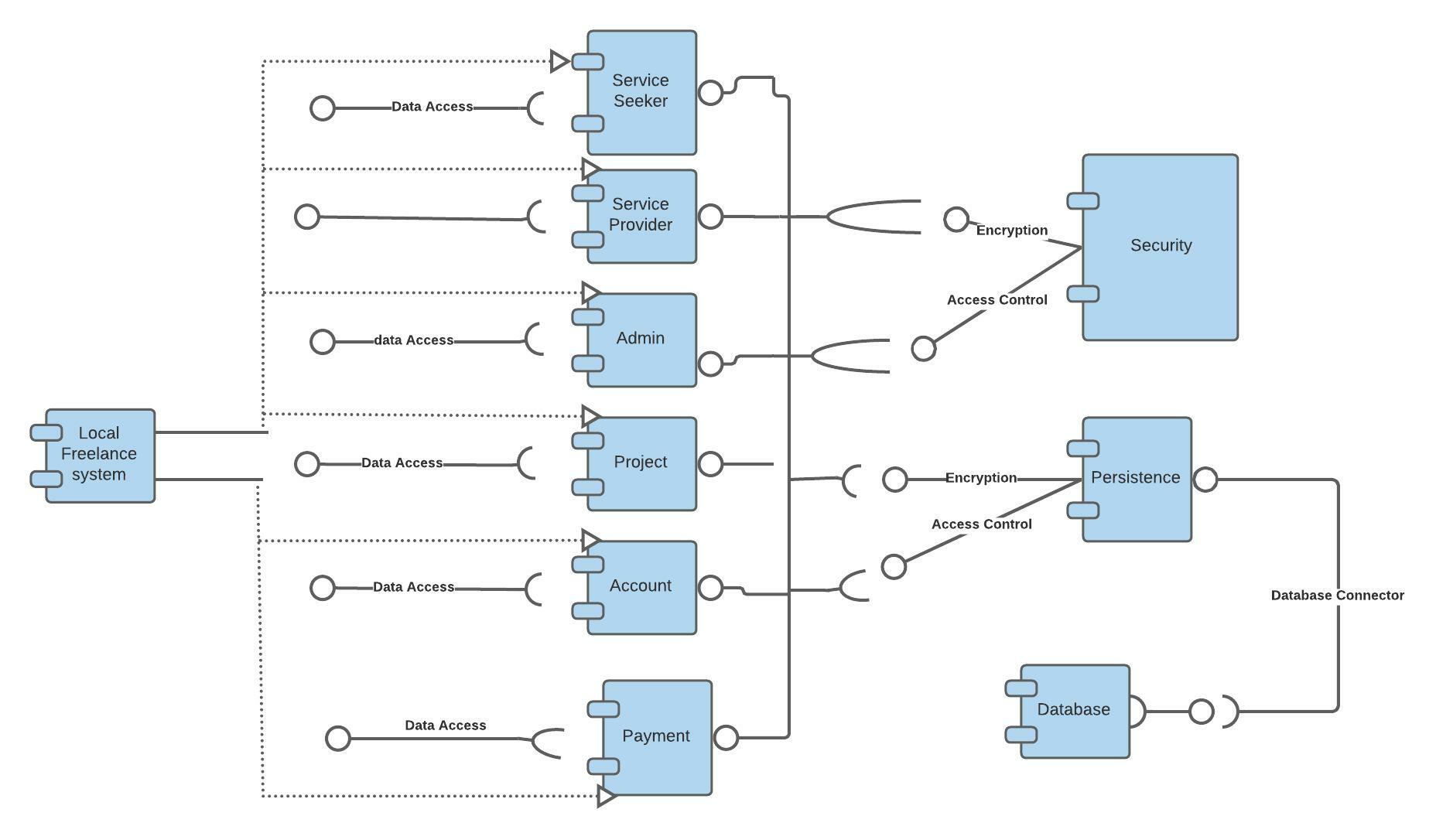


Figure 8:componet diagram

## **5.7 Deployment Diagram**

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.

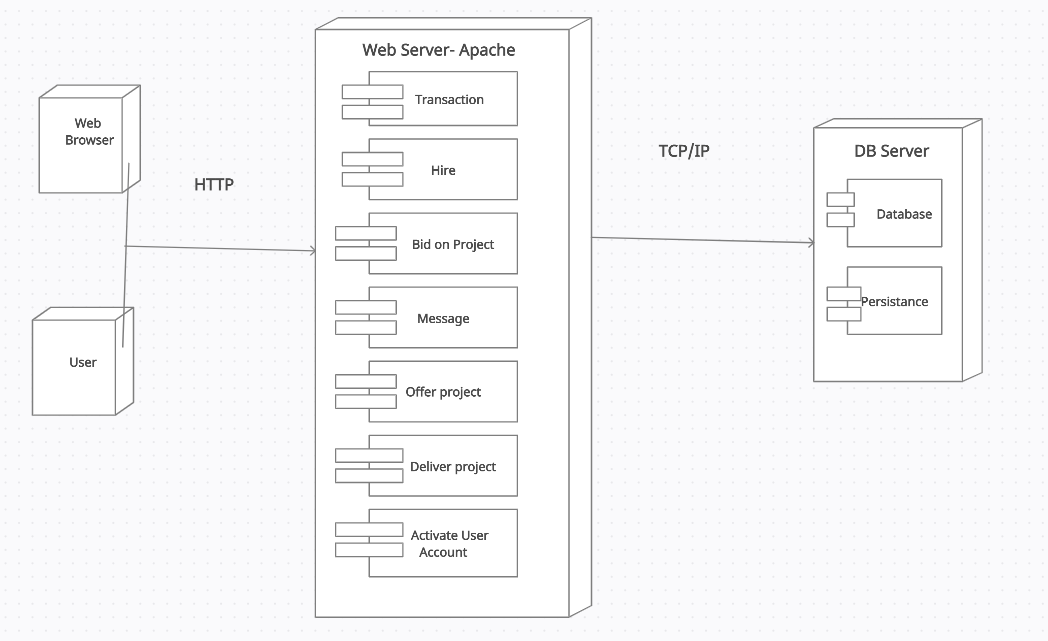


Figure 9:Deployment Diagram

## **5.8. Relational Persistent Model**

Persistence models are used to design the schema of the database. Persistence model is created whenever relational database is used to store objects and relational database is used as a mechanism to object persistence. In persistence model, class is conceptually the same as the table of relational database and attributes are the same as table columns.

The following steps convey the steps needed to convert a class diagram to entity relationship diagram.

1. Mapping Classes to TablesThe development team created a table that corresponds to a class in the mobile  
   enhanced location based drug store service, Except for the inheritance hierarchies,  
   which needs a special strategy to map.
2. Mapping Attributes To Columns

We converted simple attribute in to columns and Complex attribute if any into aseparate table since relational model is not set-valued.

1. Mapping Relationships into foreign keys

Under persistence modeling, the mapping of association results in creating a foreign key in one or more tables in the set of relational tables.

The relationships may be

* + Mapping one-to-one relationships from the class diagram.  
    There are three possibilities the relationship can be:
* Mandatory at both ends- in this case the relations can be combined
* Mandatory at one end and optional at the other: in this case, the primary key of the optional side is posted as a foreign key on the mandatory side.
* Optional at both ends: in this case, we can post the primary key of any one side toanother as a foreign key. However, choice is made wisely to identify which of the  
  two optional sides is more optional.
  + Mapping one-to-many relationships from the class diagram. In one-to-many relationships, the primary key on the one side of the relationship is  
    added to the many side as a foreign key.
  + Mapping many-to-many relationships from the class diagram.  
    For the case of Many- to- many relationships, a third relation is created and the primary keys of the two relations are posted as foreign keys to the new table. Attributes created during the relationship are also posted into this new table.

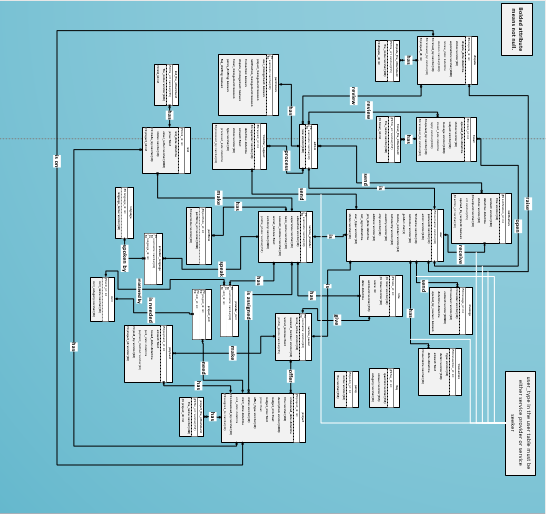


Figure 10:ER diagram

## **5.9. User Interface**

### **5.9.1. User Interface Flow Diagram**

User interface flow diagram enable us to model the high-level relationships between major user interface elements. The following depicts the user interface flow diagram for the local freelance system.

Web Based Application User Interface Flow Diagram: -

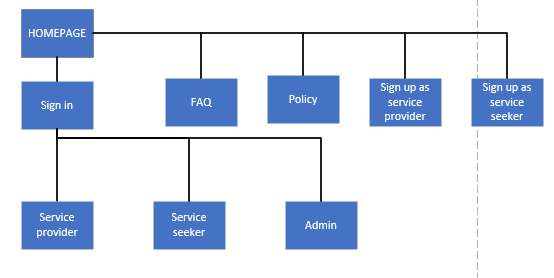


Figure 11:user Interface flow diagram

### **5.9.2. User Interface Design**

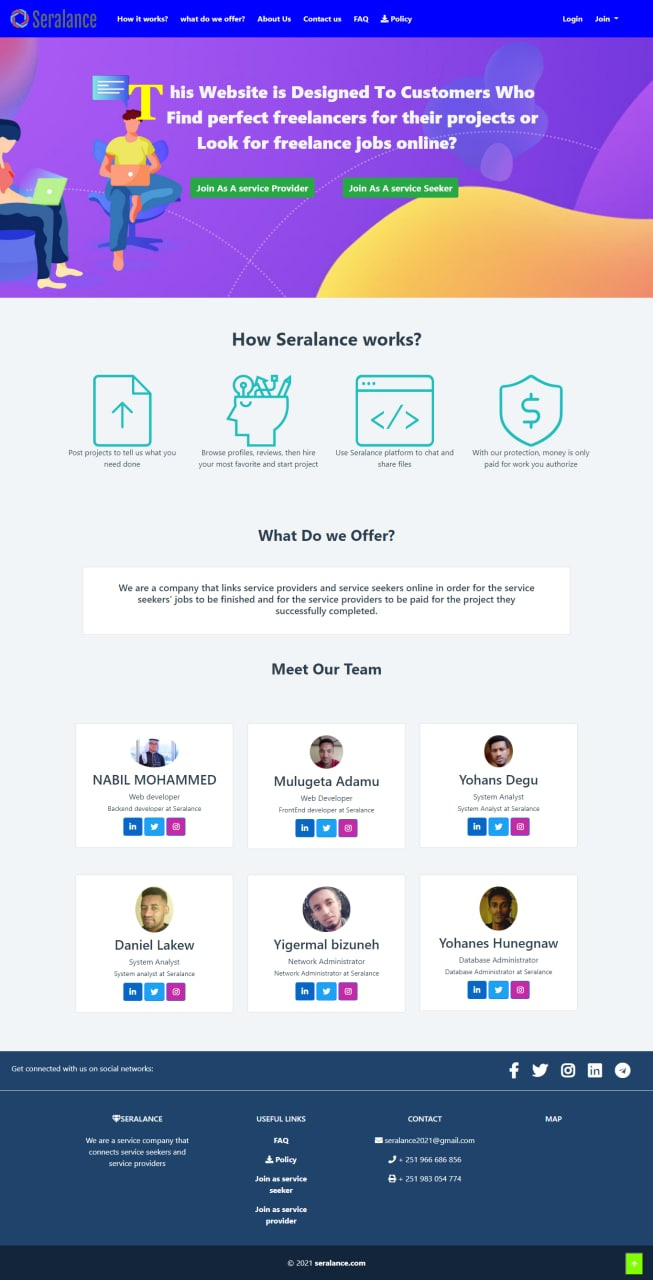


Figure 12:Landing page

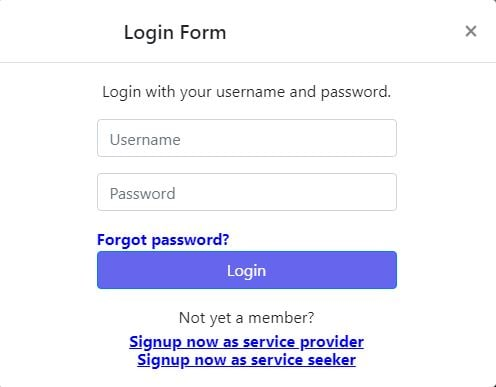


Figure 13:Login page

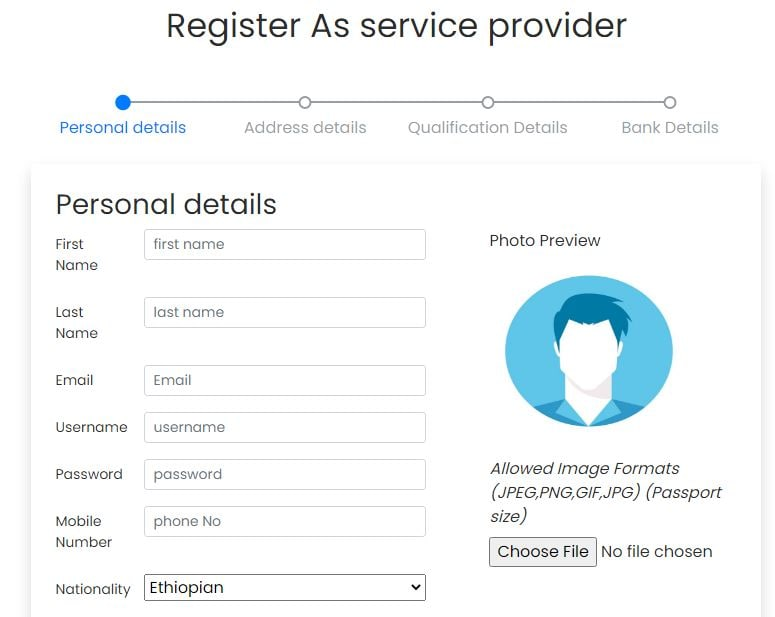


Figure 14:Service provider sign up page

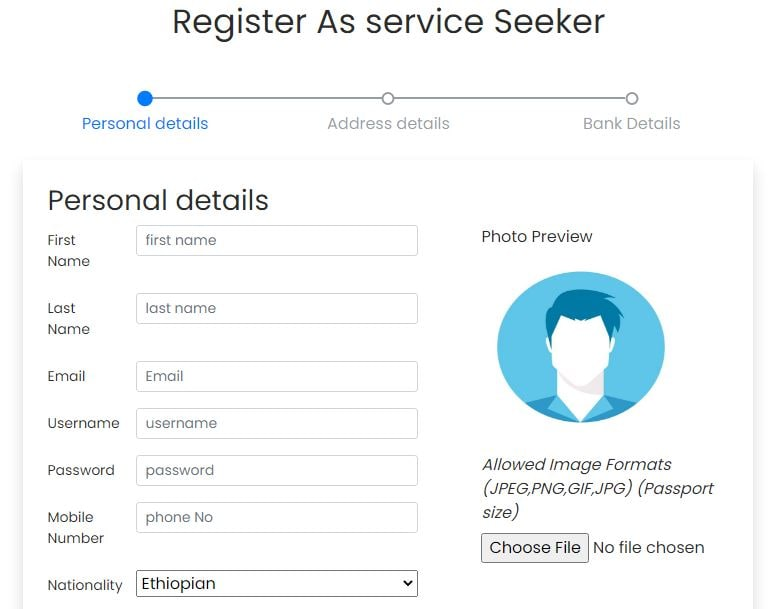


Figure 15:Service seeker sign up page

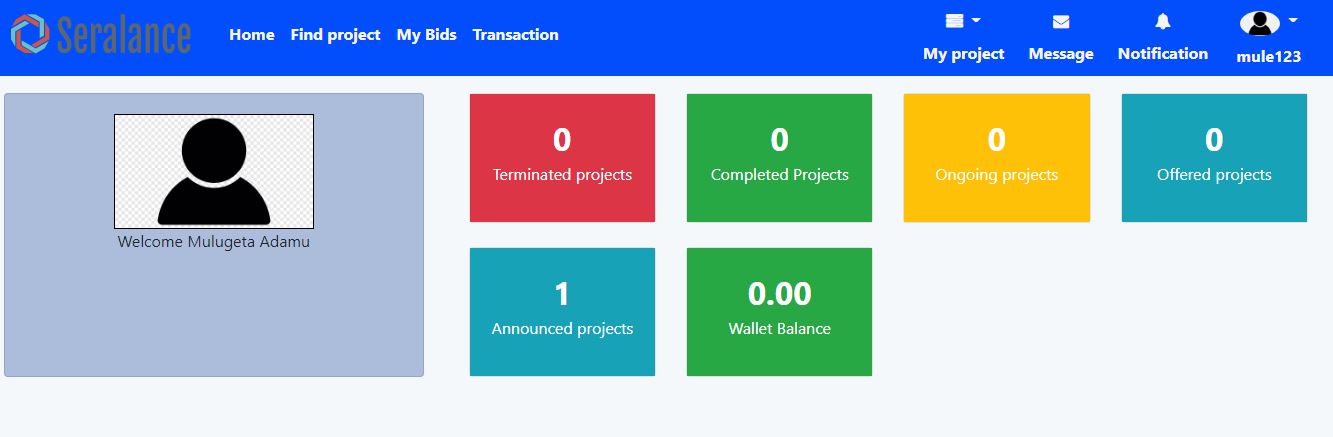


Figure 16:Home page for service provider

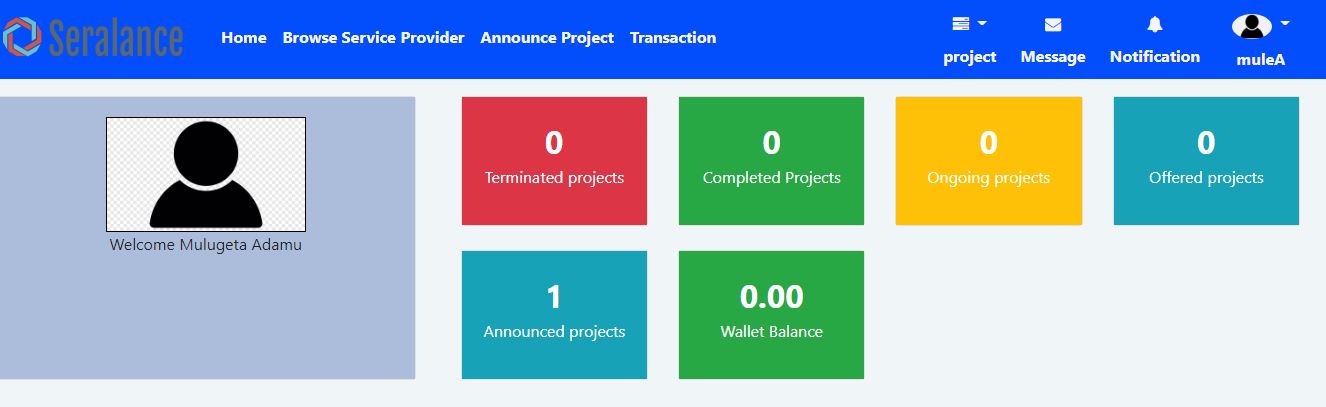


Figure 17:home page service seeker

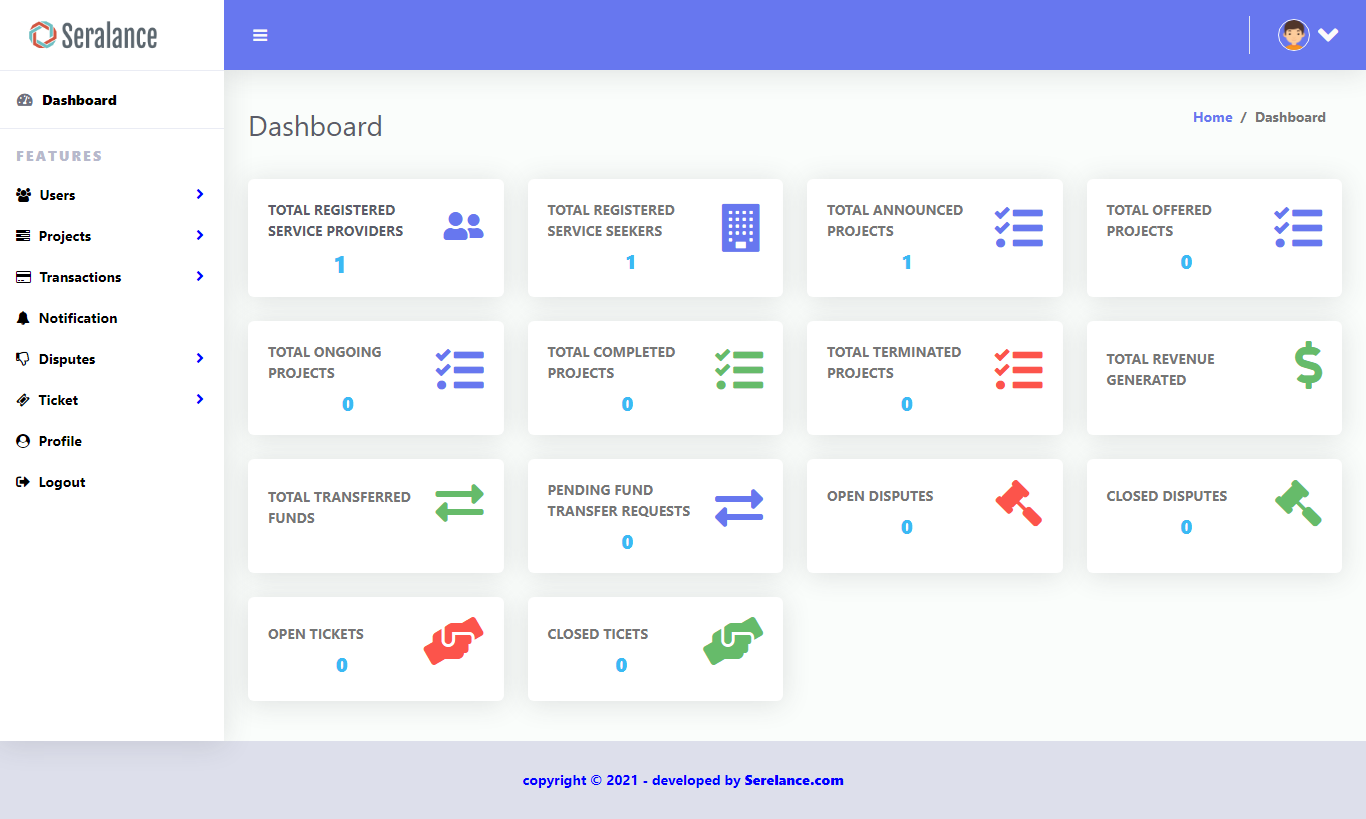


Figure 18:Home page for Admin

# **CHAPTER 6**

# **6. OBJECT ORIENTED IMPLEMENTATION**

## **6.1 Introduction**

Every system developed needs to be implemented and tested in order to be used by the end user. Implementation in the system includes implementing the attributes and methods of each object and integrating all the objects in the system, to function as a single system the implementation activity spans the gap between the detailed objects designed model and a complete of source code file that can be compiled together.

This section of the document will go indepth on the implementation of the project and it strictly follows the object oriented design shown in the previous section. This strict approach helps the team implement the project quickly, efficiently and in an orderly coordinated manner according to the designed plan.

The indepth implementation section contains the implementation technology which shows what software and hardware systems are used for the development, the testing procedures which explains how the various types of testing techniques are used to make sure the system is fault tolerant, reliable, available, secure and fulfill all other necessary requirements. The section also includes the deployment process that shows what necessary tools and technologies are needed by users to run the system efficiently.

## **6.2 Implementation Technology**

According to the technologies mentioned by the project team; the implementation of our Seralance system is done using HTML, CSS, JavaScript, JQuery and bootstrap for front end development and PHP and SQL for back end development.

## **6.4 Testing and testing procedure**

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce a quality product (Rajkumar, 2021).

### 6.4.1 Unit Testing

Unit tests is a program module that performs a specific function that can be tested. The purpose of a unit test is to ensure that the module or program performs its function as defined in the program specification. Unit testing is performed after the programmer has developed and tested the code and believes it to be error free. These tests are based strictly on the program specification and may discover errors resulting from the programmer’s misinterpretation of the specifications. Unit tests are often conducted by the systems analyst or, sometimes, by the programmer who developed the unit.

There are two approaches to unit testing. These are black-box and white-box. Black box testing is the most commonly used. In this case, the test plan is developed directly from the program specification. Each item in the program specification becomes a test, and several test cases are developed for it. White-box testing is reserved for special circumstances in which the tester wants to review the actual program code, usually when complexity is high.

### **6.4.2 Integration Testing**

Integration tests assess whether a set of modules or program that must work together do so without error. They ensure that the interfaces and linkages between difference parts of the system work properly. At this point, the modules have passed their individual unit tests, so the focus now is on the flow of control among modules and on the data exchanged among them. Integration testing follows the same general procedures as unit testing: the tester develops a test plan that has a series of tests. Integration testing is often done by a set of programmers and/ or system analysts [2].

This testing was used after unit testing and before system testing. It had helped the team to verify functional performance and reliability requirement places on major design items.

### **6.4.3 System Testing**

**System testing** is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system [3].

1. Example of Sign Up Test Case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Data | Expected Results | Output | Status  (pass/fail) |
| TC-01 | Check Sign up with password that include minimum 8 character(caps lock, number and special symbol) | Password= “John@6737” | The user successful Register to the system | The user successful Register to the system | Pass |
| TC-02 | Check the password limit when enter value less than minimum (8) | Jo@345 | The system show validation message | The system should show validation message | Pass |
| TC-03 | Check user should register by not filling the required fields | First name=” ”, email=” ”,password=” ”,account number=” ” and skill=” ” | The system should show this field is the required (red color text) on mandatory fields | The system show mandatory symbol on mandatory fields | Pass |
| TC-04 | Check sign up with valid emails | [Adamu.mulugeta@yahoo.com](mailto:Adamu.mulugeta@yahoo.com)  yohansdegu.2010@gmail.com | The system should not show validation message | The system should not show validation message | Pass |
| TC-05 | Check signup with invalid emails text(without @ symbol, dot in email) | Mulergmail.com  mulerg@mailcom | The system show validation message for valid emails | The system should show validation message for valid email | Pass |

Table 1 : sign up test case

1. Example of Login Test Case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Data | Expected  Results | Actual  Results | Status (Pass/Fail) |
| TC-06 | Check Login with valid username and password combination | Username =deguyohans2010@gmail.com Password = Joo@1234 | The system successful accept user and display user page | Successful login | Pass |
| TC-07 | Check Login with invalid username and password | Username= joo@gmail.com Password = joo123456 | The system display invalid username and password | The system display invalid user name or password | Pass |
| TC-08 | Check if the password is entered in encrypted | Username =jo@gmail.com Password = jo1234 | The password is entered in encrypted form | The password is entered in encrypted form | Pass |

Table 2: login test case

1. Example of Announce Project Test Case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Data | Expected  Results | Actual  Results | Status (Pass/Fail) |
| TC-09 | Check project announce by service seeker | Name=video editor  Description= film video | The service seeker should post project to the system | The project successfully posted | Pass |

Table 3: announce project test case

## **6.5 Deployment/Installation process**

### **6.5.1 Hardware Acquisition**

Hardware acquisition deals with specification of all the hardware at the server and client side implementation. We have identified the following specifications for our system:

* Personal Computer with
* 8GB RAM
* 50 GB hard disk

### **6.5.2 Software Acquisition**

Software plays a very crucial role in the development of any system.

* XAMPP server

# **CHAPTER 7**

## 7. **Conclusion and Recommendations**

## **7.1. Conclusion**

This project aims to create a local freelance system that allows service seekers to have their jobs done and service providers to be paid for the projects they performed successfully.

To develop this system, the project team first gathered requirements and conducted a complete business analysis of Freelancer.com's and up work's business processes. During requirement collecting, we used introspection and literature reviews, as well as numerous business analysis techniques to completely grasp the issues with the current style of operation. Based on the information we gathered we proposed a suitable solution to solve the current problem.

The project's limitation is that it does not track project work progress once it has been launched and agreed upon by the service seeker and service provider. In conclusion, despite the challenges we encountered and the project's limitations, we attempted to create an easy-to-use and useful application for our intended users.

In conclusion, even if we faced challenges during this project and despite the limitations of the project we tried to develop easy to use and functional application to our intended users.

## **7.2. Recommendation**

# This project has endeavored to implement website that helps service provider to be paid and service seeker to make their job done. During the design and development of this system, we have encountered two major opportunities that can enhance the system incredibly. In our analysis, we found out that one of the major problems of the existing system was false personal information.

# We recommend legal identification ID should be used in order to detect false personal information.

# **Reference**

[1] Souri, Alireza & Shariffloo, Mohammad & Norouzi, Monire. (2011). Formalizing class diagram in UML. 10.1109/ICSESS.2011.5982368.

[2] Alan Dennis, Barbara haley Wixom, Roberta M.Roth. System analysis and design. Indiana university, university of Virginia, university of norther lowa.

[3] Thomas Hamilton, 2021.what is system testing.

# **Appendex**

Sample code

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1 ">

<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css'>

<link href="https://fonts.googleapis.com/css?family=Poppins:400,600&display=swap" rel="stylesheet">

<link rel="stylesheet" href="../../assets/css/guest/signup.css"> <script>

document.title="Seralance- Service Seeker Sign up";

</script>

<style>

.col-form-label-sm {

padding-top: calc(.25rem + 1px);

padding-bottom: calc(.25rem + 1px);

font-size: .875rem;

line-height: 1.5;

font-weight: 900;

;

}

.form-control {

display: block;

width: 100%;

height: calc(2.25rem + 2px);

padding: .375rem .75rem;

font-size: 1rem;

line-height: 1.5;

color: #495057;

background-color: #fff;

background-clip: padding-box;

border: 2px solid #ced4da;

border-color: blueviolet;

border-radius: .93rem;

transition: border-color .15s ease-in-out,box-shadow .15s ease-in-out;

}

#profiledisplay{

border-radius: 50%;

}

.errormessage {

color: red;

font-size: 10px;

}

</style>

</head>

<body style="background-color:#dff7d0">

<?php

require\_once('../app/controllers/main.php');

$mainController = new Controller\Main();

$firstName = $lastName = $email = $username = $mobileNumber = $city = $address = $accountNumber = "";

$firstNameErr = $lastNameErr = $emailErr = $usernameErr = $passwordErr = $mobileNumberErr = $nationalityErr = $genderErr = $profilePhotoErr

= $countryErr = $cityErr = $addressErr

= $bankNameErr = $accountNumberErr = "";

if ($\_SERVER["REQUEST\_METHOD"] === "POST" && isset($\_POST['signup\_btn'])) {

$feedback = $mainController->validateSsSignup($\_POST, $\_FILES);

if ($feedback['valid'] == false) {

// setting inserted data

if (!empty($feedback['data']['firstname'])) {

$firstName = $feedback['data']['firstname'];

}

if (!empty($feedback['data']['lastname'])) {

$lastName = $feedback['data']['lastname'];

}

if (!empty($feedback['data']['email'])) {

$email = $feedback['data']['email'];

}

if (!empty($feedback['data']['username'])) {

$username = $feedback['data']['username'];

}

if (!empty($feedback['data']['mobilenumber'])) {

$mobileNumber = $feedback['data']['mobilenumber'];

}

if (!empty($feedback['data']['city'])) {

$city = $feedback['data']['city'];

}

if (!empty($feedback['data']['address'])) {

$address = $feedback['data']['address'];

}

if (!empty($feedback['data']['accountnumber'])) {

$accountNumber = $feedback['data']['accountnumber'];

}

// Setting error values

if (!empty($feedback['error']['firstname'])) {

$firstNameErr = $feedback['error']['firstname'];

}

if (!empty($feedback['error']['lastname'])) {

$lastNameErr = $feedback['error']['lastname'];

}

if (!empty($feedback['error']['email'])) {

$emailErr = $feedback['error']['email'];

}

if (!empty($feedback['error']['username'])) {

$usernameErr = $feedback['error']['username'];

}

if (!empty($feedback['error']['password'])) {

$passwordErr = $feedback['error']['password'];

}

if (!empty($feedback['error']['mobilenumber'])) {

$mobileNumberErr = $feedback['error']['mobilenumber'];

}

if (!empty($feedback['error']['nationality'])) {

$nationalityErr = $feedback['error']['nationality'];

}

if (!empty($feedback['error']['gender'])) {

$genderErr = $feedback['error']['gender'];

}

if (!empty($feedback['error']['profilephoto'])) {

$profilePhotoErr = $feedback['error']['profilephoto'];

}

if (!empty($feedback['error']['country'])) {

$countryErr = $feedback['error']['country'];

}

if (!empty($feedback['error']['city'])) {

$cityErr = $feedback['error']['city'];

}

if (!empty($feedback['error']['address'])) {

$addressErr = $feedback['error']['address'];

}

if (!empty($feedback['error']['bankname'])) {

$bankNameErr = $feedback['error']['bankname'];

}

if (!empty($feedback['error']['accountnumber'])) {

$accountNumberErr = $feedback['error']['accountnumber'];

}

}

}

?>