Payroll taxes system

Table of Contents

[1. Introduction 1](#_Toc532104400)

[1.1 Objective 1](#_Toc532104401)

[1.2 Document Conventions 1](#_Toc532104402)

[1.3 Intended Audience 1](#_Toc532104403)

[2. Requirements 2](#_Toc532104404)

[2.1 User Requirements 2](#_Toc532104405)

[2.2 Systems Requirements 2](#_Toc532104406)

[2.3 Functional Requirements 2](#_Toc532104407)

[2.4 Nonfunctional Requirements 2](#_Toc532104408)

[2.4.1 Product requirements 2](#_Toc532104409)

[2.4.2 Organisational requirements 2](#_Toc532104410)

[2.4.3 External requirements 2](#_Toc532104411)

[3. System modeling 3](#_Toc532104412)

[3.1 Use case diagram 3](#_Toc532104413)

[3.2 Sequence diagram 3](#_Toc532104414)

[3.3 Class diagram 3](#_Toc532104415)

[4. Architectural design 3](#_Toc532104416)

[4.1 Say Layered architecture 3](#_Toc532104417)

[4.2 Say Client-sever architecture 4](#_Toc532104418)

[5. Implementation 4](#_Toc532104419)

[5.1 GUI of … 4](#_Toc532104420)

[5.2 Screenshot of … 4](#_Toc532104421)

[6. Testing 4](#_Toc532104422)

[6.1 Test cases for object classes 4](#_Toc532104423)

[7. Discussion 5](#_Toc532104424)

**List of figures**

[Figure 1: Use case diagram of .. 3](#_Toc532104467)

[Figure 2: Use case diagram of ... 3](#_Toc532104468)

[Figure 3: Use case diagram of .... 3](#_Toc532104469)

[Figure 4: Sequence diagram of ... 3](#_Toc532104470)

[Figure 5: Class diagram of ... 3](#_Toc532104471)

[Figure 6: Layered architecture of ... 3](#_Toc532104472)

[Figure 7: Client-sever architecture of ... 4](#_Toc532104473)

[Figure 8: Graphical interface of ... 4](#_Toc532104474)

[Figure 9: Screenshot of … 4](#_Toc532104475)

[Figure 10: Screenshot of ... 4](#_Toc532104476)

[Figure 11: Screenshot of ... 4](#_Toc532104477)

[Figure 12: Screenshot of ... 4](#_Toc532104478)

[Figure 13: Screenshot of ... 4](#_Toc532104479)

[Figure 14: Test cases of 4](#_Toc532104480)

# 1-Introduction

## Objective

We are trying to provide a more easy taxes calculation by more technology rather than traditional ways. We trying to provide suitable GUI to make it easier to the user to interact with the system. We are trying to avoid computational errors that may happen by the traditional ways.

## Document Conventions

* verification

1. Safety
2. Security
3. Availability
4. Efficiency
5. dependability

* priority

1. High :-( efficiency -- maintainability--safety)
2. Medium :-( acceptability -- availability)

## Intended Audience:-

This project is for the Payroll system and it is deals with employers. This project is useful for the clients and many people.

# 2. Requirements

**2.1 User Requirements:**

1. Provide an acceptable and easy to use interface
2. Provide two different systems in the calculations based on the city and its system.
3. Each employee must have a special account on the site.

**2.2 Systems Requirements**

1. Should provide information on how to use the system to the user.
2. Should provide choices about when the user pays and the nearest city to him.
3. Should contain registration and login to ensure security.
4. Should redirect the user to the same page if the information he entered is wrong.
5. Should calculate taxes according to the user’s choices.
6. Should provide details about how the tax was calculated.

**2.3 Functional Requirements**

1. The user logs in to process his or her account for taxes.
2. The system introduces an introduction to how to use it, then log in or register to us the system to calculate his or hertax.
3. The system should not deduct more than 50% of employees pay in tax.
4. System checks the user’s inputs to verify the validity of all data entered to ensure reliable results.
5. System allows a user to enter the data for the previous salary by a percentage not exceeding 30%.
6. Work is broken into a two paths in caseUK rate (for England, Wales and Northern Ireland)and the second caseScottish rate.
7. The tax value is calculated in each case from the previous according to the tableandB or C or D1 or D2.

**2.4 Nonfunctional Requirements**

### 2.4.1Product requirements

**2.4.1.1Safety Requirements**

1. Data is backed upon another server to avoid loss in the event of any problem.

There is no single point of failure.

**2.4.1.2Security Requirements**

1. The system cannot be accessed by anyone unless he has an account.
2. It is impossible for any customer to view the customer's data, since each customer has his / her own account, unless he / she personally gives his / her password to another.

**2.4.1.3 Software Quality Attributes**

1. **Availability**: The system is functionally working with multiple user needs and operating successfully with less hardware needs and available 24 hour.
2. **Flexibility**: The application will be able to change easily in response to multiple user needs and many different categories.
3. **Maintainability**: The upcoming versions will fix the bugs; if found in the previous ones.
4. **Testability**: The application will be tested several times before being into production.
5. **Supportability**:The application offers an information in the home page.
6. **Efficiency**: The application meets user needs despite the lowest possibilities of hardware.
7. **Usability**: The system’s pages will be easy to understand.

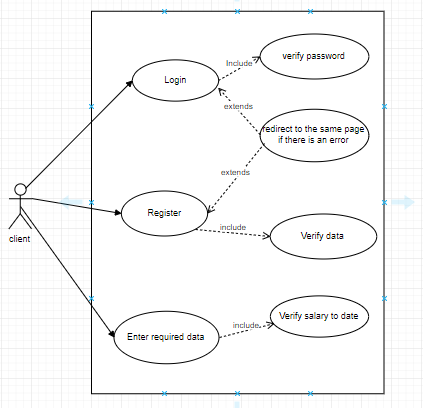
**2.4.2****External requirements**

1. The system is subject to legal records agreed by the competent authorities.
2. The system is constrained by the Scottish and UK rules for calculating the taxes
3. There is a mutual trust between the client and the employees of the System who have developed it so that they are assured of the results provided by the system.
4. The system performs its tasks by a systematically so that it is easy for the customer to calculate his taxes.

3. **System modeling**

**3.1 Use case diagram**

**Figure 1: use case diagram of client**



|  |  |
| --- | --- |
| Condition | Action |
| If the client enters the password right | It is verified and then directed to calculation page. |
| If email or password are wrong | The client will be redirected to the same to enter them again. |

Table 1: tabular description of login

Table 2: tabular description of Register

|  |  |
| --- | --- |
| **Conditions** | Actions |
| If the client enters the information required to register correctly | The information are then verified and directed to the calculation page. |
| If the client enters wrong information | The client will be redirected to the same page to enter the information again. |

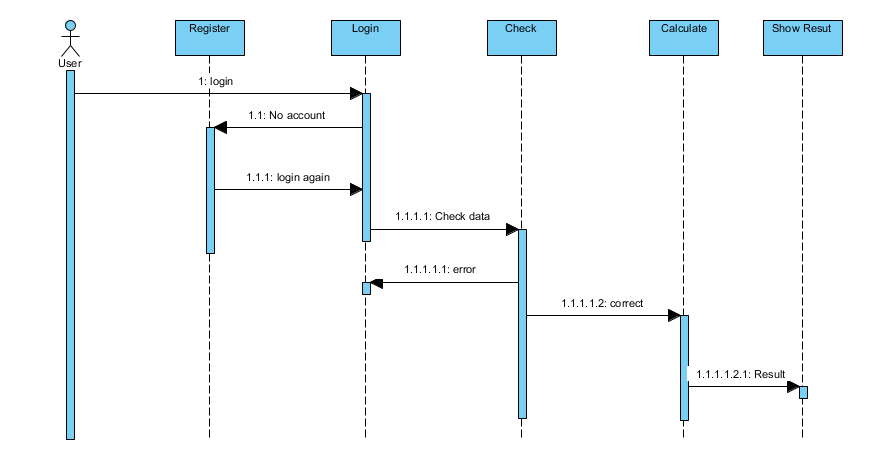
Table 3: tabular description of codes

|  |  |
| --- | --- |
| Condition | ACTION |
| If code is br and pay method is weekly or monthly | Multiply the total taxable pay by 0.20 |
| If code is do and pay method is weekly or monthly | Multiply the total taxable pay by 0.40 |
| If code is d1 and pay method is weekly or monthly | Multiply the total taxable pay by 0.45 |

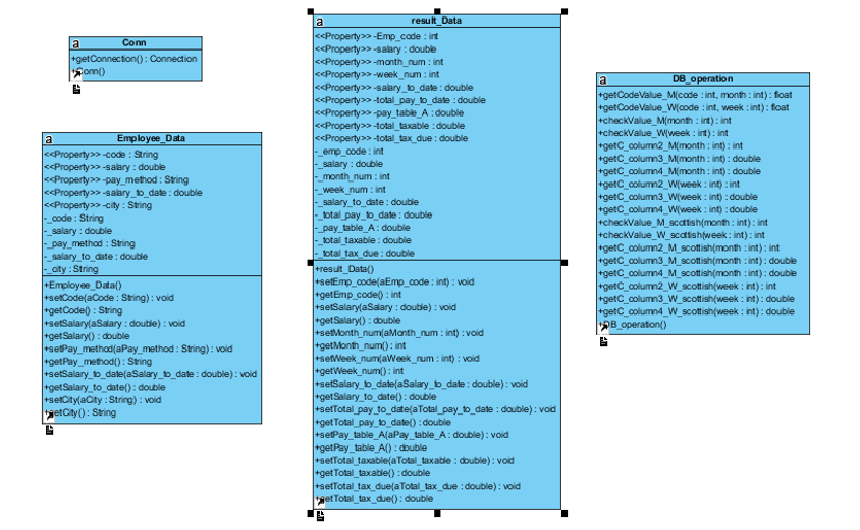
|  |  |
| --- | --- |
| If client chooses Cardiff, Chelsea, Liverpool, London,  Manchester | Tax is calculated according to UK rate |
| If client chooses Aberdeen, Ayr,  Edinburgh, Elgin, Ullapool | Tax is calculated according to Scottish rate |

**Table 4: Tabular description of the rate**

**3.2 Sequence diagram**

****

**3.3 Class diagram**

****

**4. Architectural design**

* **Client Server Architecture**:

Tables server

Tables data

User server

User data

System

Client4

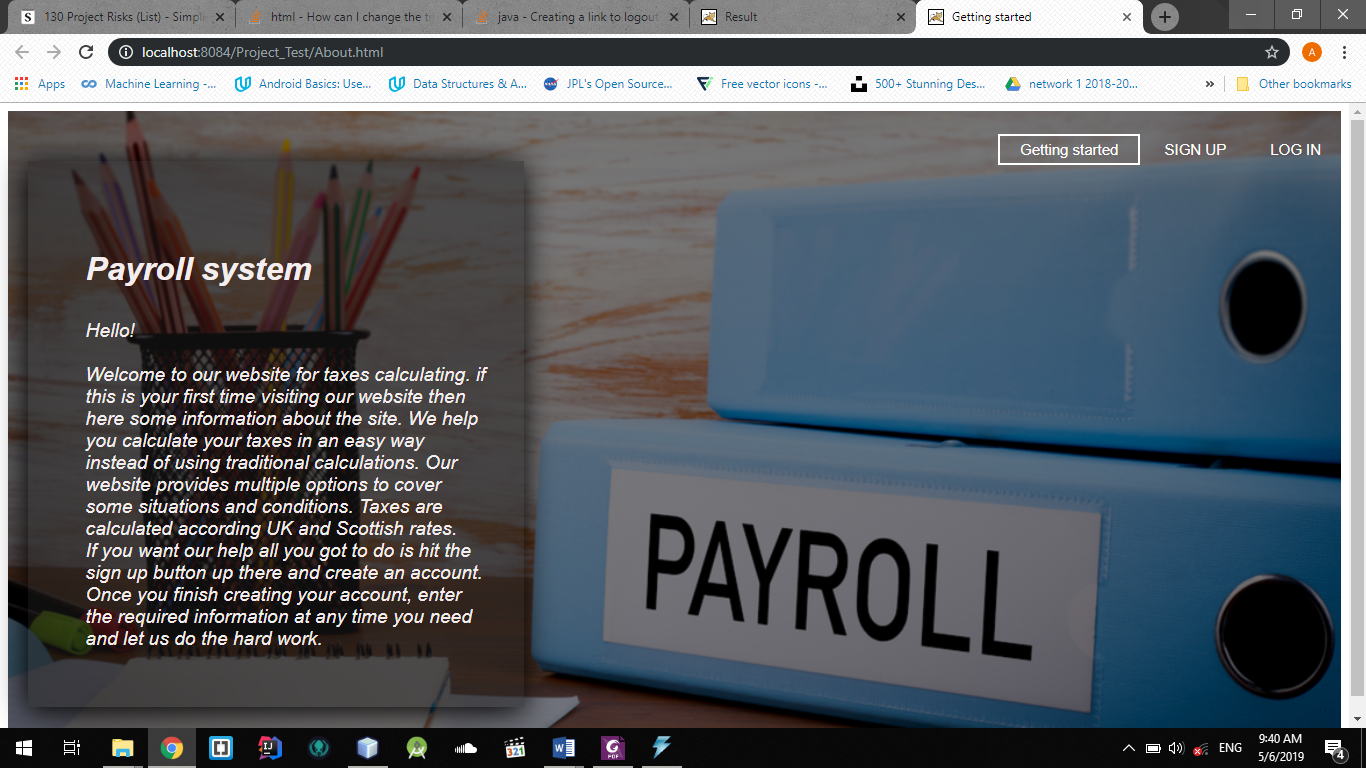
Client3

Client2

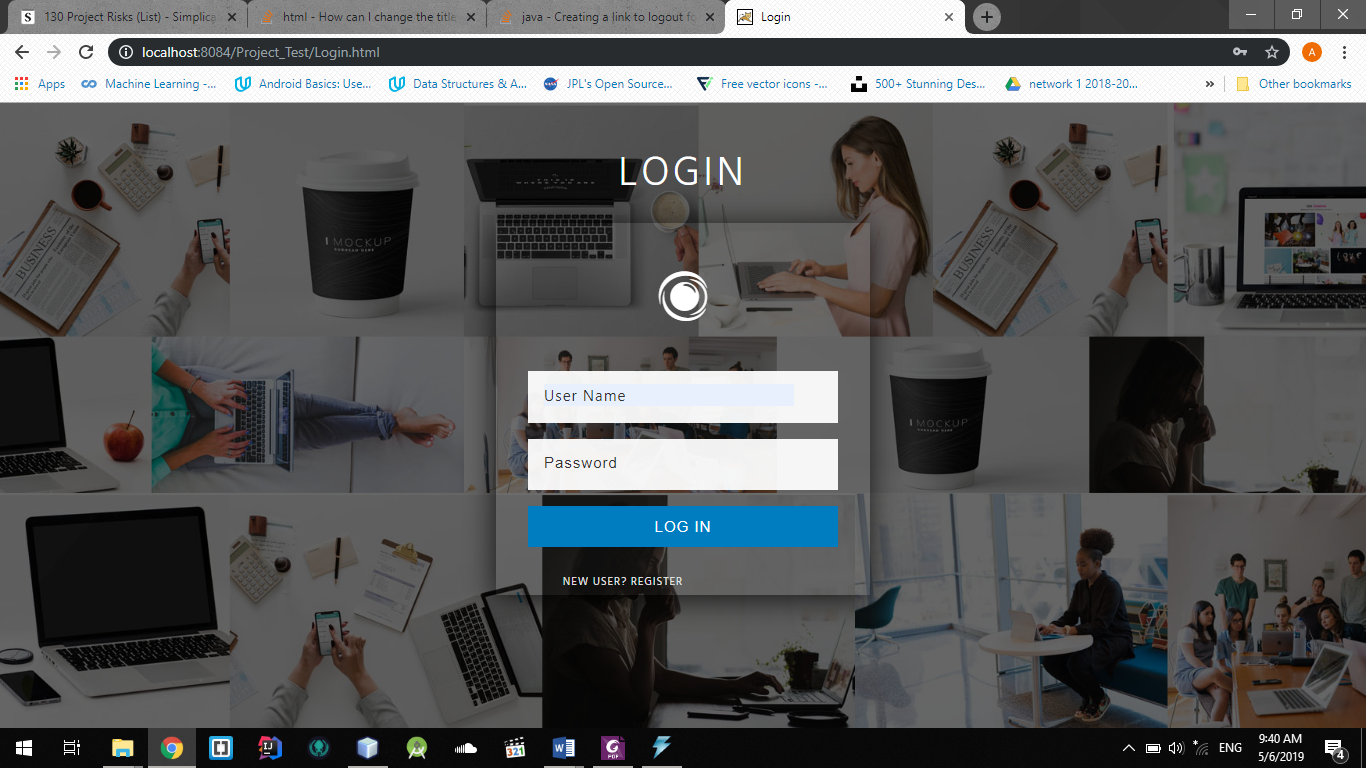
Client1

**5. Implementation**

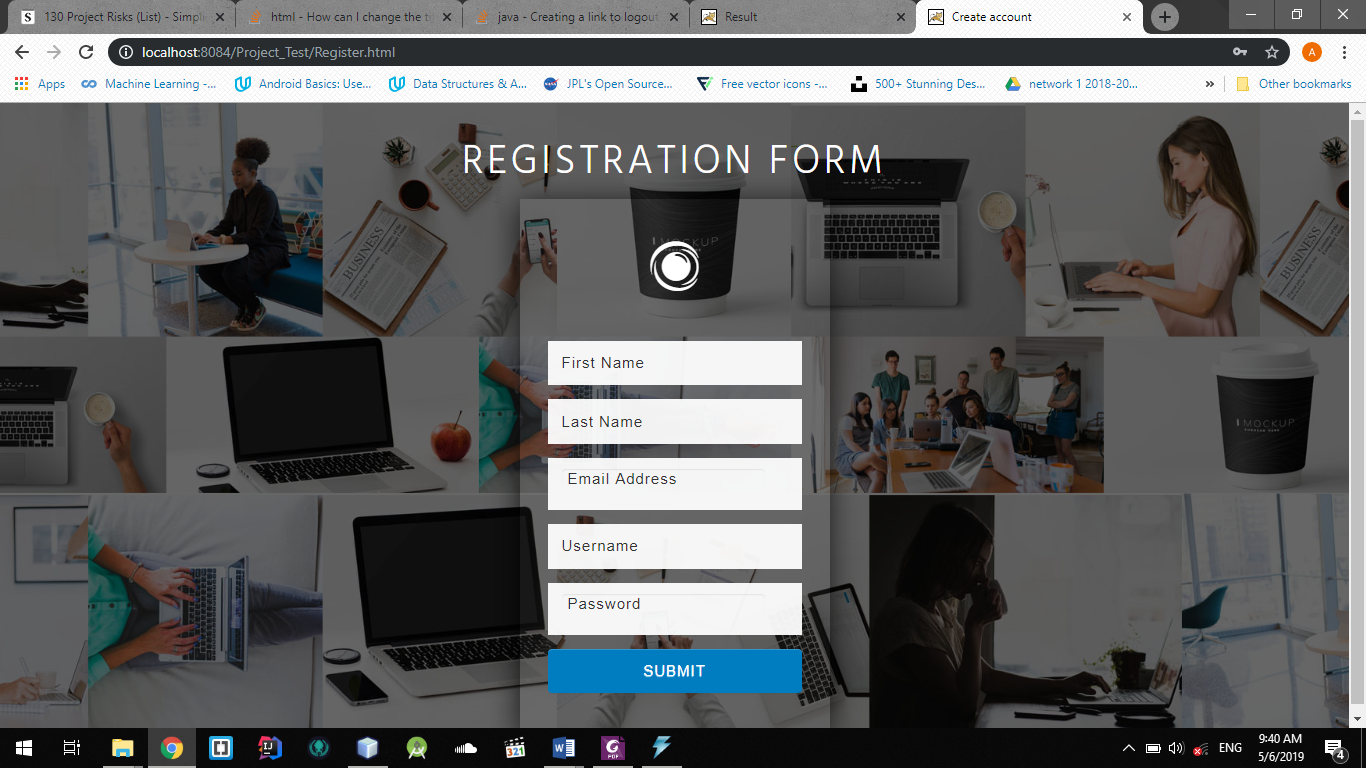
5.1 GUIof html page of the Home page

****

**5.2** Screenshot of login page



**5.4** Screenshot of registrationpage



**5.2** Screenshot of calculation page

