

Favours4Neighbours

The development of a web application to deliver household help to those in need of a particular skillset.

A thesis submitted for the

*Higher Diploma in Software Development* Supervised by Mary Ryan

Submitted to the Limerick Institute of Technology, *2nd-May-2021*

Niamh Egan

K00106199

## Abstract – Draft

Due to a global health pandemic in the spread of COVID-19 societies most vulnerable people were now at risk of contracting a highly contagious a deadly disease. The elderly and those with serious underlying health conditions had to be cut off from society, with family members abroad unable to return and inter county travel restrictions in place. Nursing homes were badly affected accounting for 1,500 deaths from COVID-19. A persons inability to complete small household tasks is often a driving factor in deciding to move to a care facility.

This application aims to connect those who need small jobs done around the house, shopping collected, smoke alarms fitted etc. with people in the neighborhood willing and available to help out. Family members abroad unable to return to an elderly relative in rural Ireland can ensure that small tasks are carried out and the household maintained so that their loved ones can continue to live at home and enjoy their independence.

## Acknowledgements

## Table of Contents

[Abstract – Draft 2](#_Toc67140016)

[Acknowledgements 3](#_Toc67140017)

[Table of Contents 4](#_Toc67140018)

[List of Figures 5](#_Toc67140019)

[Introduction 6](#_Toc67140020)

[Research Chapter 7](#_Toc67140021)

[Software Problem 7](#_Toc67140022)

[Technologies 9](#_Toc67140023)

[Design Chapter 11](#_Toc67140024)

[Introduction 11](#_Toc67140025)

[Requirements Specification 11](#_Toc67140026)

[Use Cases Favours4Neighbours 18](#_Toc67140027)

[Use Case Diagram 30](#_Toc67140028)

[Wireframes 31](#_Toc67140029)

[Screen Dump 37](#_Toc67140030)

[Activity Diagram 38](#_Toc67140031)

[Enhanced Entity Relationship Design Diagram 39](#_Toc67140032)

[MVC 40](#_Toc67140033)

[Logical Architecture Diagram 42](#_Toc67140034)

[Implementation 43](#_Toc67140035)

[Introduction 43](#_Toc67140036)

[Burn down chart 44](#_Toc67140037)

[Screen shots of functionality 45](#_Toc67140038)

[Screen shots of database 46](#_Toc67140039)

[Class implementation code 47](#_Toc67140040)

[Conclusion 48](#_Toc67140041)

[Bibliography 49](#_Toc67140042)

## List of Figures

## Introduction

## Research Chapter

### Software Problem

Covid-19 has had a detrimental impact on societies ability to socialize, see friends, visit neighbors with our most venerable confined to their homes. Many people have experienced loneliness and isolation as a result of quarantine and lock downs, and this has detrimental mental health impact as people lose connection and interaction with each other. (Tzung-Jeng Hwang, 2020). The Favours4Neighbours web application aims to re-establish connections with neighbors in your locality that may need a small job done with people willing and available to lend a hand.

According to AgeAction an Irish Charity which provides services to the elderly people of Ireland, a major factor in deciding to move to residential care, particularly for those older people in the unfortunate position of not having family members calling is something as simple as not being able to change a light bulb, install a smoke alarm, trim a hedge etc.

In 2015 Alone Charity launched its “Home First Campaign – Why are we forcing older people out of their homes?” In Ireland, more than one third of older people in long-term nursing homes beds have low/ medium dependency needs, meaning that they could live at home if they were be provided with adequate home care. The percentage of our older population in nursing home care is already 35% greater than the EU average and the average stay in Irish nursing homes is 1.6 years longer than our UK neighbours.  These figures prove that we are forcing older people into nursing homes before they need to go in.  (ALONE, 2015).

Unlike other European Countries where more integrated models of older care services and supports exist, the model of care for elderly people remains underdeveloped in Ireland. This is reflected in limited intermediate/ or step-down options for our older population. As a result significant demands continue to be placed on the nursing home sector to meet the care needs of our older population. This has consequential cost implications for the individual, their families and the State. It also has a serious impact on the acute hospital sector, as people who cannot access nursing home care will, in many cases, be left with no option but to seek acute hospital care. Not only does this have significant cost implications for the State, it also impacts on the ability of the entire population to access acute care. (BDO, 2014)

Favour4Neighbours can help bridge the gap by matching people in need of these small jobs done with people willing and able to complete them, enabling people to get the support they need to stay int their own homes.

Favours4Neighbours could also assist households who have seen changes in their routine since the pandemic stated. Studies suggest that domestic outsourcing reduces the time spent on housework as well as the gender gap (e.g., Craig et al. 2016; Raz-Yurovich & Marx 2019; also see Hank 1998). Outsourcing should save time spent on the routine and time-intensive tasks that often fall on women’s shoulders (S. Bianchi, 200).

The COVID-19 pandemic, however, limited households’ opportunities for outsourcing. This might have disadvantaged women, particularly highly-skilled ones, who have been shown to be the main beneficiaries of domestic outsourcing. Along similar lines, mothers may also be likely to carry the main burden of new or regained responsibilities resulting from closures of childcare facilities and schools (e.g., Bujard et al. 2020; Kohlrausch & Zucco 2020; also see Schulz & Engelhardt 2017).

Favours4Neighbours can help individuals struggling to maintain full time work from home, and managing home schooling by getting their shopping collected, dry cleaning collected etc

### Technologies

#### Web Applications

(Standing, 2001) describes web applications as being multi-functional applications that typically encompass more stakeholders than traditional desktop applications. These stakeholders can include amongst others, internal stakeholders such as employees and external stakeholders such as customers and suppliers. A web application was chosen for this project for its accessibility to the target audience as there is no need to download an application.

#### HTML

HTML (Hyper Text Mark-up Language) has quickly become the standard language for the web, with the latest version (HTML5) the foundation for web application development. Some of the most significant upgrades in HTML5 include; simplicity; storage; media; forms and mobile support.

#### PHP

PHP (Hypertext Pre-processor) is a popular scripting language designed for web development. It is a procedural object-oriented language. PHP code can be embedded within the HTML page and will be executed each time the page is visited and interpreted by the application server, thus creating dynamic output. PHP was chosen for its performance; scalability; built-in libraries; and ease of learning (Welling & Thomson, 2008).

#### Composer (Package manger)

Composer is a package manager for PHP programming language. I was chosen for its management of dependencies and the provision of the required libraries to manage the application development. Version 2.0 was used for the development of this application.

#### Code Igniter 4

CodeIgniter is an Application Development Framework to assist in building web applications using PHP. Code Igniter 4 was chosen for its rapid development framework as it provides a set of libraries for commonly needed tasks and a simple interface with logical structure to access these libraries while minimizing the amount of code needed to product for a given task making debugging easier.

#### Visual Studio Code

Visual studio code was chosen to build the application as it is an open source code editor with built in Git which was used for version control of the application

#### GitHub

GitHub is a version management tool used in Industry and is used for Favours4Neighbours as means of version control and backup. As the project advanced commits were backed up to the Git repository to keep track of work done via Visual Studio Code.

#### MySQL

MySQL is an open source Relational Database Management System (RDBMS) that facilitates the creation, retrieval, storage and updating of data through the Structured Query Language (SQL) standard. It provides concurrency control protocols to enable secure multi-user database access. Some of the many strengths of MySQL as a RDBMS are; performance; reliability; low cost; ease of use; portability; support. When MySQL is used in conjunction with PHP, it creates the foundation for interactive and dynamic web applications (Nixon R. , 2012)

## Design Chapter

### Introduction

### Requirements Specification

1. Administrator Functions
   1. Login

**Type:** Menu Item

**Location:** Main landing page (not logged in users)

**Action:** Administrator will provide a username and password to login. These are verified by implementing a DB lookup. If successful the user will be able to view the Administrator Home Page. If unsuccessful – the login page will be presented again with a warning of remaining login attempts. User is permitted 3 login attempts. After 3 failed attempts the users account is locked.

* 1. Logout

**Type:** Menu Item

**Location:** All Administrator pages

**Action:** When selected from the Administrator home page – this function will log the user out. All cookies and sessions will be deleted.

* 1. User Administration

**Type:** Menu Item

**Location:** All Administrator pages except User Administration

**Action:** When selected the User Administration page will open and the following navigation menu items will appear to enable performance of the following functions for User Administration

* + 1. View Registered Users

**Type:** Menu Item

**Location:** All User Administration

**Action:** When selected this menu item will present a table of all user IDs and usernames along with status (active/blocked).

* + 1. Search for User Data by user Name:

**Type:** Menu Item

**Location:** All User Administration

**Action:** When selected this menu item will present a form which will allow an Administrator to enter a user ID. When the form is submitted – the system will respond with either a message stating ‘Invalid User ID’ or with a table containing the user’s details – ID, Name, Username along with buttons to block/activate a user and reset the user’s password.

* + 1. Suspend User Access:

**Type:** Button

**Location:** All User Administration

**Action:** As described in 1.3.2 – a button will be provided on the individual user details result table which will allow the Administrator to make a user inactive or block the user. Once blocked the user will not be able to log in to the system.

* + 1. Enable User

**Type:** Button

**Location:** All User Administration

**Action:** As described in 1.3.2 – a button will be provided on the individual user details result table which will allow the Administrator to make a user active or unblock the user. Once unblocked the user will be able to log in to the system.

* + 1. View all Live Chat

**Type:** Menu Item

**Location:** Administration Home Page

**Action:** This menu option will allow the Administrator to view all live chats between users.

1. Neighbour / Helping Hands Functions
   1. Login

**Type:** Menu Item

**Location:** Main landing page (not logged in users)

**Action:** User will provide a username and password to login. These are verified by implementing a DB lookup. If successful the user will be able to view the User Home Page. If unsuccessful – the login page will be presented again with a warning of remaining login attempts. User is permitted 3 login attempts. After 3 failed attempts the users account is locked.

* 1. Logout

**Type:** Menu Item

**Location:** All User pages

**Action:** When selected from the User home page – this function will log the user out. All cookies and sessions will be deleted.

* 1. User Home

**Type:** Menu Item

**Location:** All User pages

**Action:** When selected the User Home page will open and the following navigation menu items will appear to enable performance of the following functions for User Home

* + 1. Update / Edit profile

**Type:** Menu Item

**Location:** Edit Profile

**Action:** When selected the user will be able to add information to their profiles which are visible to all users.

* + 1. Add/Update Skills

**Type:** Menu Item

**Location:** Edit Profile

**Action:** When selected the user will be able to add skills to their profiles which are visible to all users these skills will be used to match with tags in a job posting to match skills with jobs requiring those skills

* 1. Create Job

**Type:** Menu Item

**Location:** User Home

**Action:** When selected the user will be able to add a description of the job that needs to be done including the date, time, skill required, machinery required to complete the job successfully.

* 1. Live Chat

**Type:** Menu Item

**Location:** All User pages

**Action:** When selected the user will be able to directly message users that are online and have applied to complete the job advertised. The user will enter a user name or id if known and a chat will be established between the users.

* 1. Search active Jobs by Status.

**Type:** Menu Item

**Location:** All User pages

**Action:** When selected this menu item will present a drop down menu which will allow an user to Select a job Status and enter a job title or skill and location When the form is submitted – the system will respond with either a list of jobs in a table containing the job title, location, skills needed, date job needs to be done, job poster details.

* + 1. Apply

**Type:** Button

**Location:** Search active Jobs by Status.

**Action:** When selected this button will submit a request to apply to complete the job advertised. A message will be sent to the users inbox who posted the job informing them who applied to do the job.

* 1. Inbox

**Type:** Menu Item

**Location:** All User pages

**Action:** When selected this item will load the inbox of the user where all messages relating to applications to complete a job will be gathered. The user can open the mail and review the application, contact the applicant and accept or reject their application to complete the job.

### Use Cases Favours4Neighbours

|  |  |  |
| --- | --- | --- |
| **Use Case No. 1** | **Participating Actor(s):**  **User / Neighbour** | **Use Case 1 of 5** |
| **Use Case Name:** Create user | | |
| **Description: This function creates a new user on the system assuming they are not already registered** | | |
| **Entry Condition(s):**   1. **A new user has been identified and their details need to be added to the system** 2. **The registration function of the system has been invoked** | | |
| **Flow of Events:**   1. **The system responds by displaying the registration form which requires the following information** 2. **The user complete the registration form to include email address, re-enter email address, name, user name, contact number and address.** 3. **The user clicks register** 4. **The form will be validated to ensure the user inputs valid information** | | |
| **Exit Condition(s):**   1. **the new user details have been added to the system** 2. **the user is redirected to the log in page** | | |
| **Alternate Flows / Exceptions Handling:**  **If the user attempts to create an account with data already registered on the database the user will be instructed to log in using their existing account, or can reset their password** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 2** | **Participating Actor(s):**  **Administrator** | **Use Case 2 of 5** |
| **Use Case Name:** Search for user by Name | | |
| **Description:**  This function allows the administrator to search for accounts for users via User Name, First and Surname | | |
| **Entry Condition(s):**  **The administrator is logged on and the system** | | |
| **Flow of Events:**   1. **The administrator logs on to the system** 2. **The administrator selects search and enters the relevant ID number of the account they are looking for** 3. **The associated account is displayed if exists** | | |
| **Exit Condition(s): the ID is not registered** | | |
| **Alternate Flows / Exceptions Handling:**  **If the ID number is incorrect or invalid the administrator can search by name, email address or phone number as another means of locating an account** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 3** | **Participating Actor(s):**  **User/Neighbour** | **Use Case 3 of 5** |
| **Use Case Name:** Create Job | | |
| **Description:** This functionality enables a user/Neighbour to create a Job the need done. | | |
| **Entry Condition(s):**  The user is logged in and has successfully navigated to the Create Job page | | |
| **Flow of Events:**   1. **The user selects Create Job from the Jobs tab in the main Navigation** 2. **The user completes selects the Job Category from a drop down list** 3. **The user adds a Job Title and Description** 4. **The user adds a date the job needs to be completed** 5. **The user selects whether the equipment will be provided or if the applicant should have their own** 6. **The user specifies whether they job will be paid and if so how much** | | |
| **Exit Condition(s): The user selects Save to save the job created** | | |
| **Alternate Flows / Exceptions Handling:**   1. **The date must be input using the Calendar or date format DD/MM/YYYY** 2. **Fields marked by \* must be completed to save the job** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 4** | **Participating Actor(s):**  **User/Neighbour** | **Use Case 4 of 5** |
| **Use Case Name: Delete Job** | | |
| **Description: The function allows the user to delete a task they have created** | | |
| **Entry Condition(s):**  **The user is logged in**  **The user has created the job** | | |
| **Flow of Events:**   1. **The user selects my jobs from the home page** 2. **The user selects the job they want to delete** 3. **If the job has not been assigned to someone the user can select delete to delete the task** 4. **If there is a person assigned to the task the cancel job use case applies** | | |
| **Exit Condition(s): the delete task can be saved provided there is not applicant assigned to complete the task** | | |
| **Alternate Flows / Exceptions Handling:**   1. **The Job must have no accepted applicant assigned to the job to enable deleting the task** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 5** | **Participating Actor(s):**  **User/Neighbour** | **Use Case 5 of 5** |
| **Use Case Name: Update Job** | | |
| **Description: the function enables the user to update details to a created task** | | |
| **Entry Condition(s): the user is logged in**  **The user has previously created a job** | | |
| **Flow of Events:**   1. **The user selects My Jobs from the home page** 2. **The user selects update job** 3. **The user can make changes to the job provided it is 48 hours in advance and the job category is the same** | | |
| **Exit Condition(s): the update job will be saved if the user has not made critical changes to it and it is in advance of 48 hours prior to the date specified** | | |
| **Alternate Flows / Exceptions Handling:**  **the user will need to cancel and create a new job if the scope of the job originally created is intended to be changed** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 6** | **Participating Actor(s):**  **Administrator** | **Use Case:** |
| **Use Case Name: Suspend User** | | |
| **Description: The administrator needs to suspend a user account** | | |
| **Entry Condition(s): The administrator is logged into the system**  **The Users account is active** | | |
| **Flow of Events:**   1. **The administrator searches for the user by User Name and find the account in question** 2. **The administrator selects suspend user** 3. **The administrator clicks save** | | |
| **Exit Condition(s): The system updates the users account to suspended** | | |
| **Alternate Flows / Exceptions Handling:**  **Is the user is already suspended as error message with appear stating the users account is already suspended and will give the option to enable the account or cancel the action** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 7** | **Participating Actor(s):**  **Administrator** | **Use Case:** |
| **Use Case Name: Enable User** | | |
| **Description: The administrator needs to reactivate a suspended user account** | | |
| **Entry Condition(s): The administrator is logged into the system**  **The User has been suspended.** | | |
| **Flow of Events:**   1. **The administrator searches for the user by User Name and find the account in question** 2. **The administrator selects enable user** 3. **The administrator clicks save** 4. **The system responds by enabling the user** | | |
| **Exit Condition(s): the screen returns to the search user by ID screen** | | |
| **Alternate Flows / Exceptions Handling:**  **If the users account is already enabled the enable button will be greyed out with the status of the account set to active the administrator will not be able to click enable**  **There will be an option to cancel the action and return to the search for user screen** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 8** | **Participating Actor(s):**  **User/Neighbour** | **Use Case** |
| **Use Case Name: Log In** | | |
| **Description:**  **This function enables a registered user to access the favours4neighbours web application** | | |
| **Entry Condition(s):**  **the user/ administrator will launch a browser and type in the favours4neighbours URL and hit search**  **the user is already registered on the web application**  **the user is not already logged in** | | |
| **Flow of Events:**   1. **The favours4neighbours login screen appears** 2. **The user enters their credentials** 3. **The user clicks login** 4. **The user is redirected to the homepage** | | |
| **Exit Condition(s): the user is redirected to the homepage** | | |
| **Alternate Flows / Exceptions Handling:**  **If the user is not already registered there will be an error stating details not found. The user will be instructed to either reset their credentials if they have an email address already registered or to complete the registration as a new user.** | | |

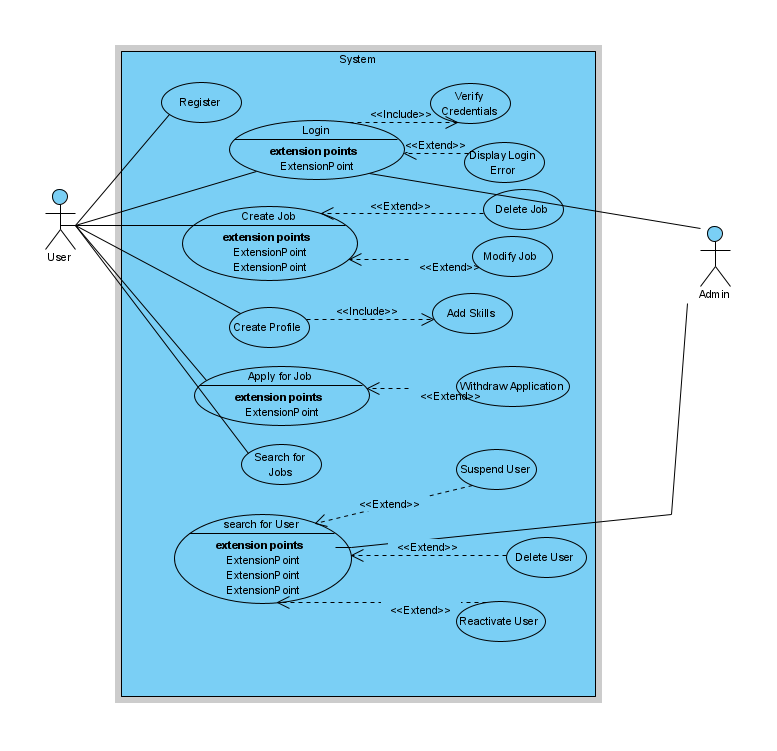
|  |  |  |
| --- | --- | --- |
| **Use Case No. 9** | **Participating Actor(s):**  **User/Neighbour** | **Use Case** |
| **Use Case Name: Log Out** | | |
| **Description: this function logs out and closes the current session of the application** | | |
| **Entry Condition(s): the user is logged in** | | |
| **Flow of Events:**   1. **The user clicks the log out button in the main navigation bar** 2. **The system displays a pop up asking the user do they want to exit the application** 3. **The user selects yes** 4. **The system logs the user out of the application** | | |
| **Exit Condition(s): The system redirects to the log in page** | | |
| **Alternate Flows / Exceptions Handling:**  **The user selects No when prompted to confirm if they want to log out, the pop up closes and the user remain on the page that they were on before initiating the log out request** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 10** | **Participating Actor(s):**  **User/Neighbour** | **Use Case** |
| **Use Case Name: Apply for Job** | | |
| **Description: this function enables a user to apply for a job on the system** | | |
| **Entry Condition(s): the user is logged in**  **there is a job created and active on the system** | | |
| **Flow of Events:**   1. **The user searches for jobs on the system** 2. **The user selects a job** 3. **The user selects apply** 4. **The system responds asking the user to confirm they want to apply for the selected job** 5. **The user selects yes** | | |
| **Exit Condition(s): the user has confirmed their intention to apply for the job and clicked ok** | | |
| **Alternate Flows / Exceptions Handling:**  **If the user selects no when asked to confirm the application they system responds by closing the pop up screen and returning the user to the job search screen** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 11** | **Participating Actor(s):**  **User/Neighbour** | **Use Case** |
| **Use Case Name: Accept Application** | | |
| **Description: this function enables a user who has created a job to accept an application to complete the job from another registered user** | | |
| **Entry Condition(s): the user is logged on**  **There is an active application made to complete the job**  **The job has been created by the user** | | |
| **Flow of Events:**   1. **The user views the notifications** 2. **The user is happy for the applicant to complete the job** 3. **The user selects accept** 4. **The system responds by asking the user to confirm to accept the application** 5. **The user selects yes** | | |
| **Exit Condition(s): the applicant has been successfully accepted to complete the job. The user whos application has been successful will receive notification.**  **The users will now be able to communicate directly** | | |
| **Alternate Flows / Exceptions Handling:** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case No. 12** | **Participating Actor(s):**  **User/Neighbour** | **Use Case** |
| **Use Case Name: Decline Application** | | |
| **Description:**  **this function enables a user who has created a job to decline an application to complete the job from another registered user** | | |
| **Entry Condition(s):**  **the user is logged on**  **There is an active application made to complete the job**  **The job has been created by the user** | | |
| **Flow of Events:**   1. **The user views the notifications** 2. **The user is not happy for the applicant to complete the job** 3. **The user selects decline** 4. **The system responds by asking the user to confirm to decline the application** 5. **The user selects yes** 6. **The system responds and declines the application and generates a message to the applicant informing them the application has not been accepted** | | |
| **Exit Condition(s): the application has been declined the applicant has been informed** | | |
| **Alternate Flows / Exceptions Handling:** | | |

### Use Case Diagram



### Wireframes

#### Common Screens

Log in and Log out allows access and exit from the application by the registered user

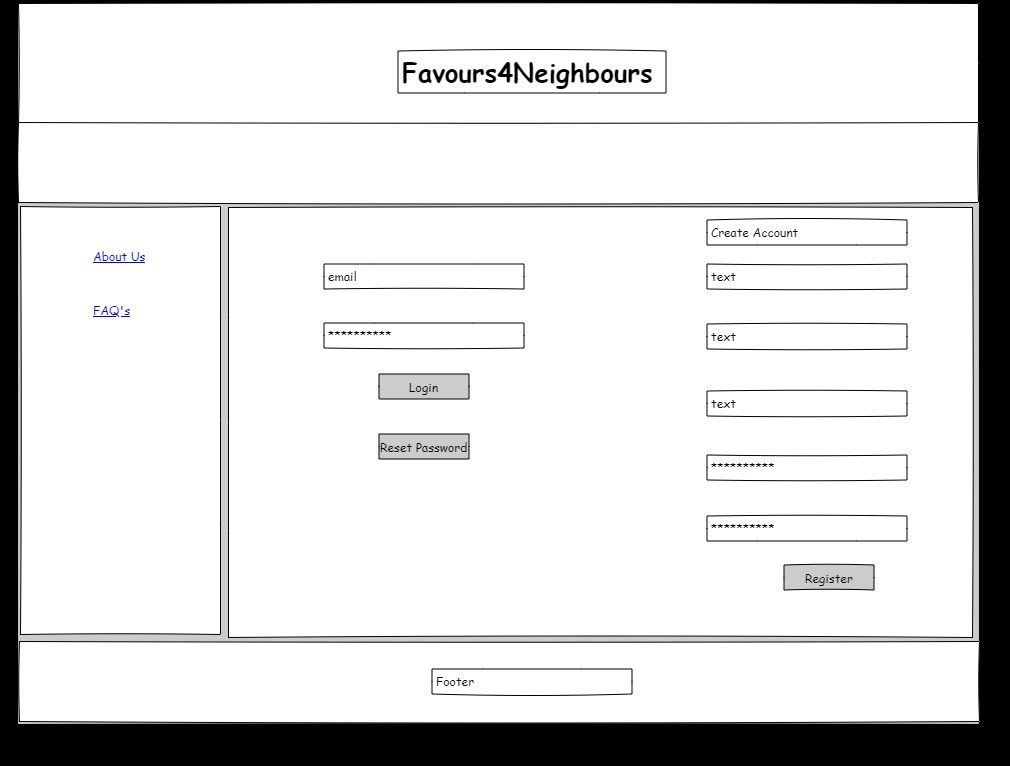


Figure Login Page

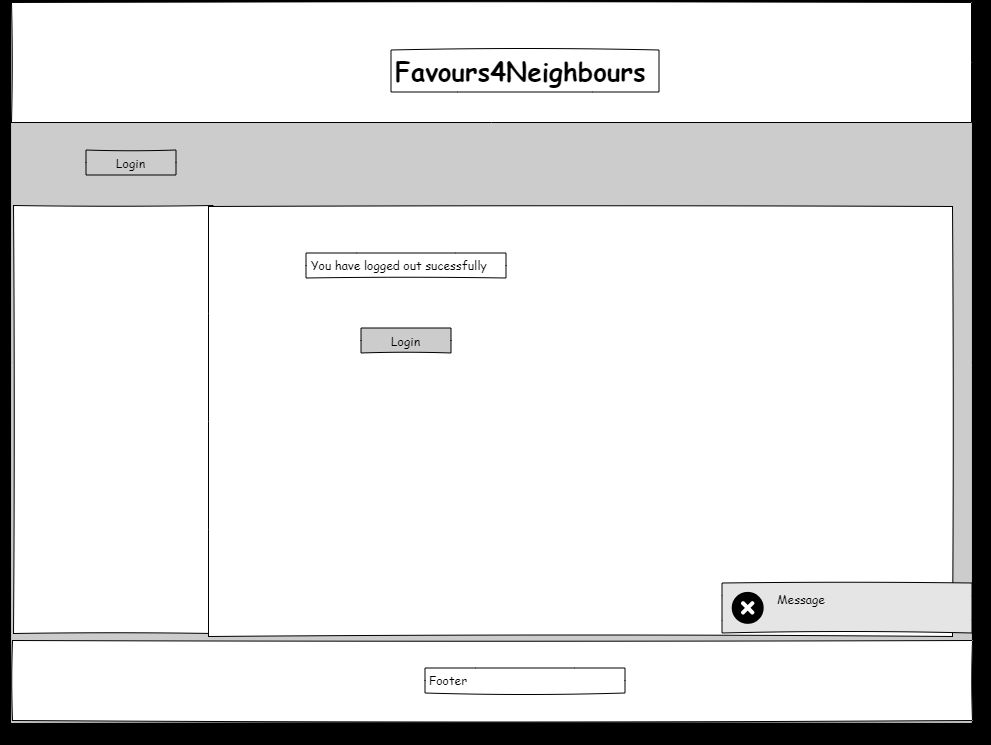


Figure Logout Screen

#### User / Neighbour Home

Home Screen view for Neighbour User shows main navigation, Profile overview, Chat, and includes Buttons to Create Job, Search Jobs and view My Jobs.

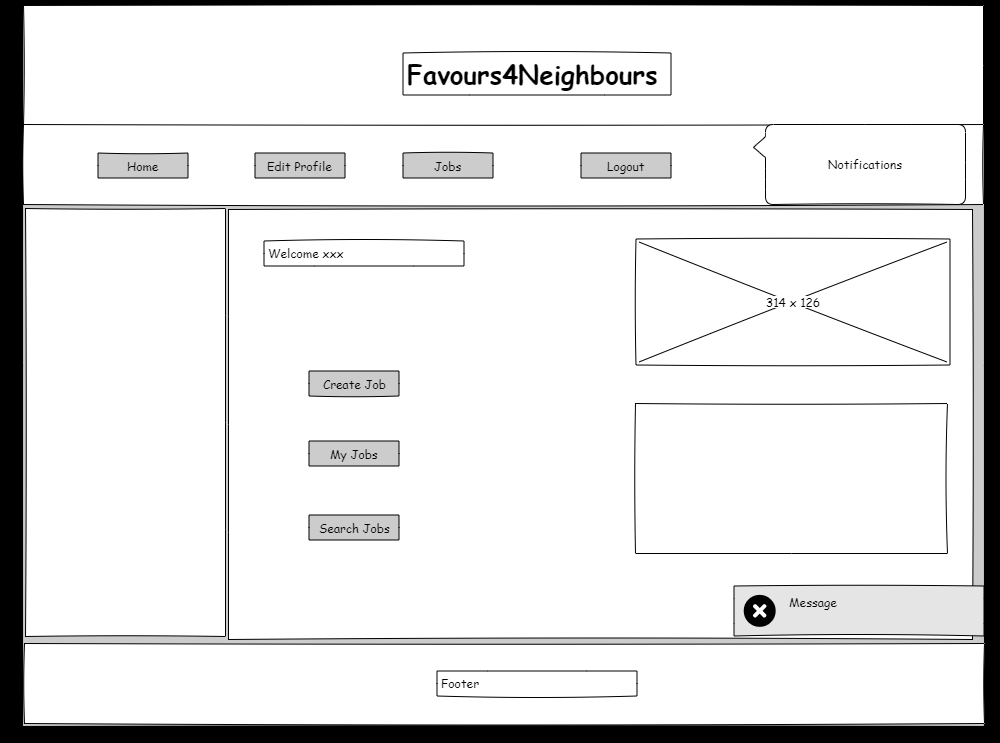


Figure User/Neighbour Home View

#### User/Neighbour Edit Profile

Edit Profile allows the user to edit the details of their profile such as the job tags, availability, etc.

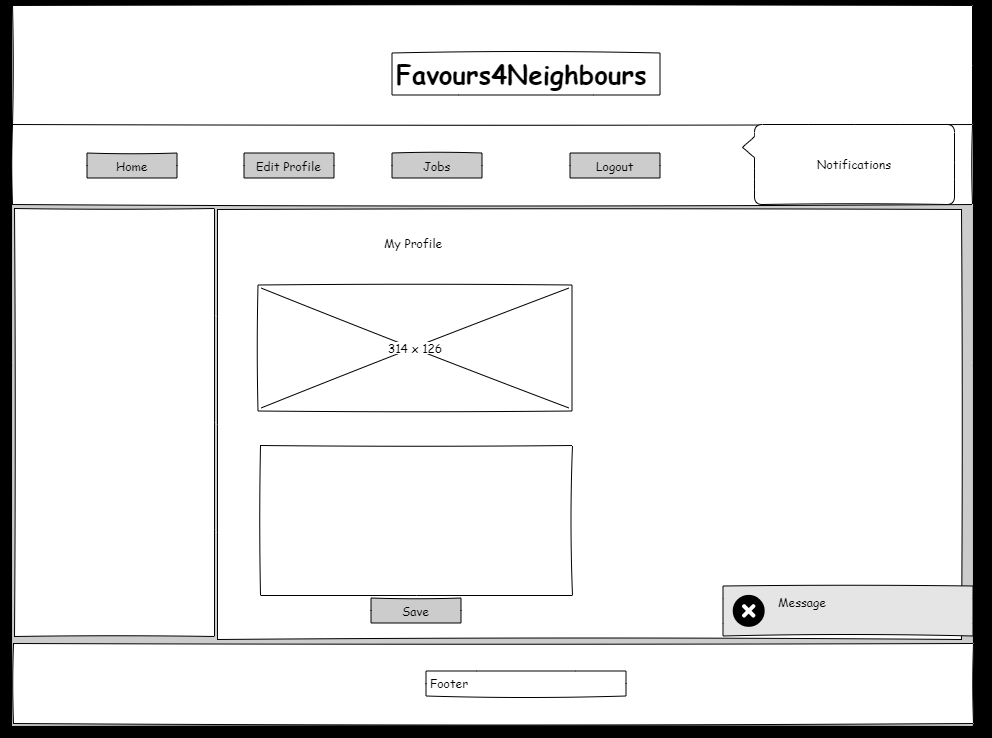


Figure User/Neighbour Edit Profile

#### User /Neighbour Jobs View

Jobs allows the user to view jobs the have created, favors they have confirmed to carry out and view applications for favors

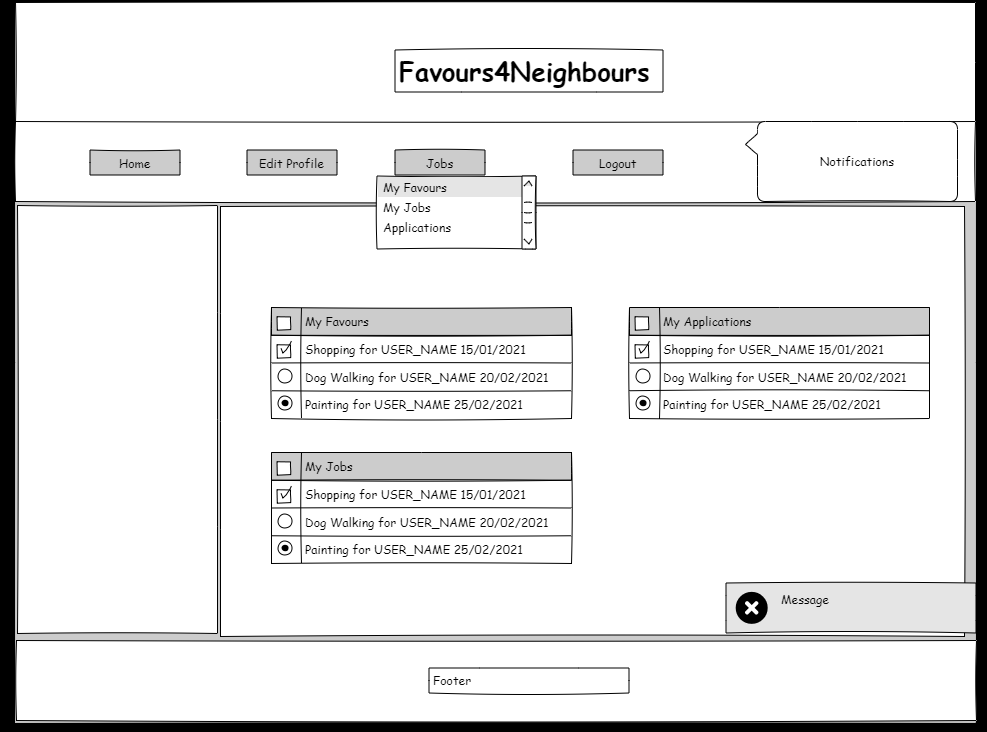
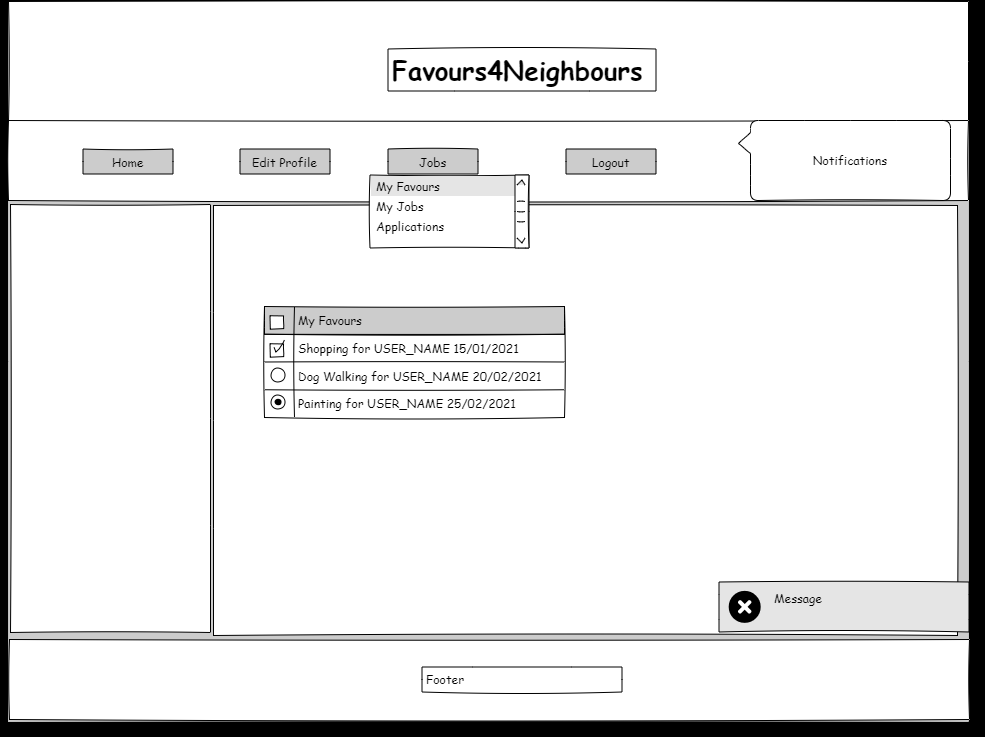


Figure User/Neighbour Jobs View

#### Neighbour Favors View

My Favors allows the user to view what favors they have signed up for



### Screen Dump

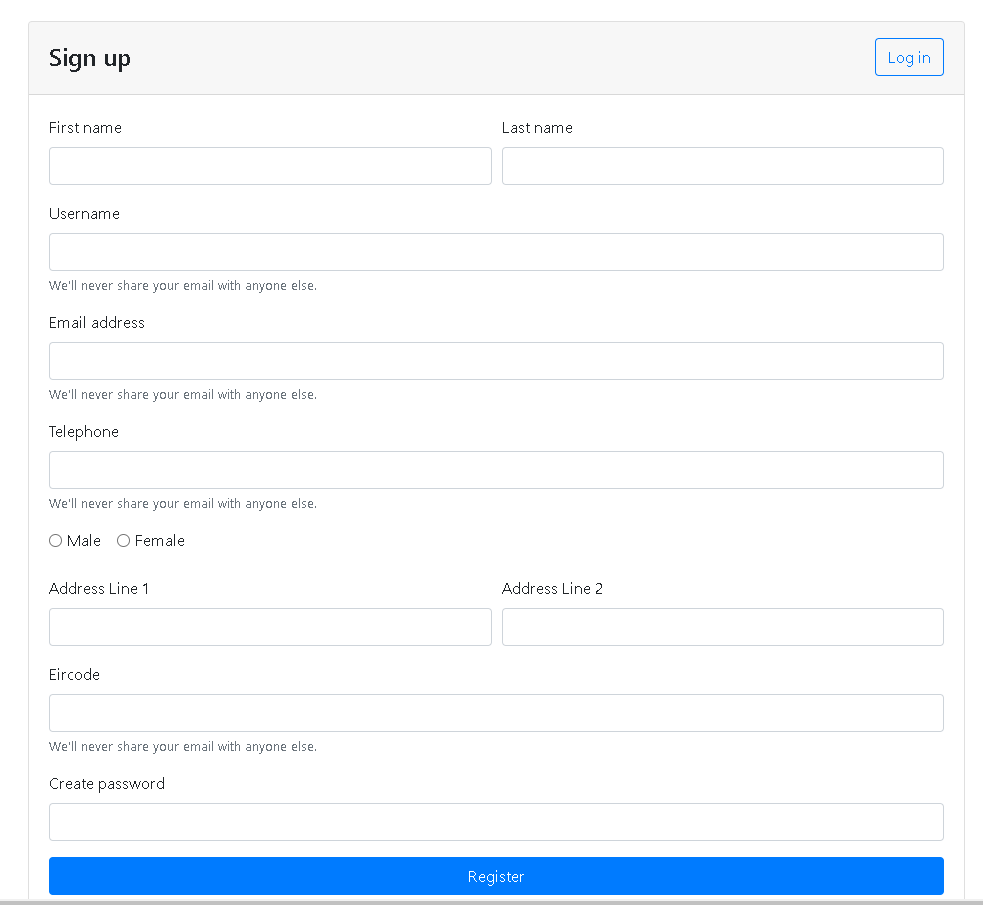


Figure Registration View

### Activity Diagram

#### Log in / Register

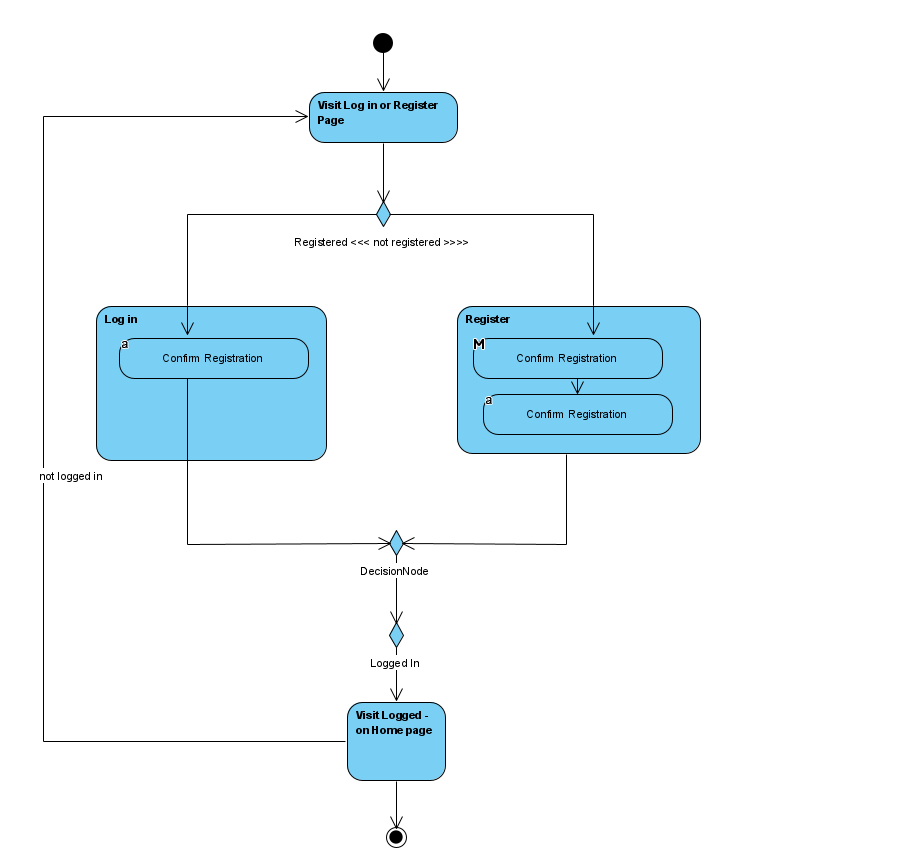


Figure Activity Diagram Log In / Register

### Enhanced Entity Relationship Design Diagram

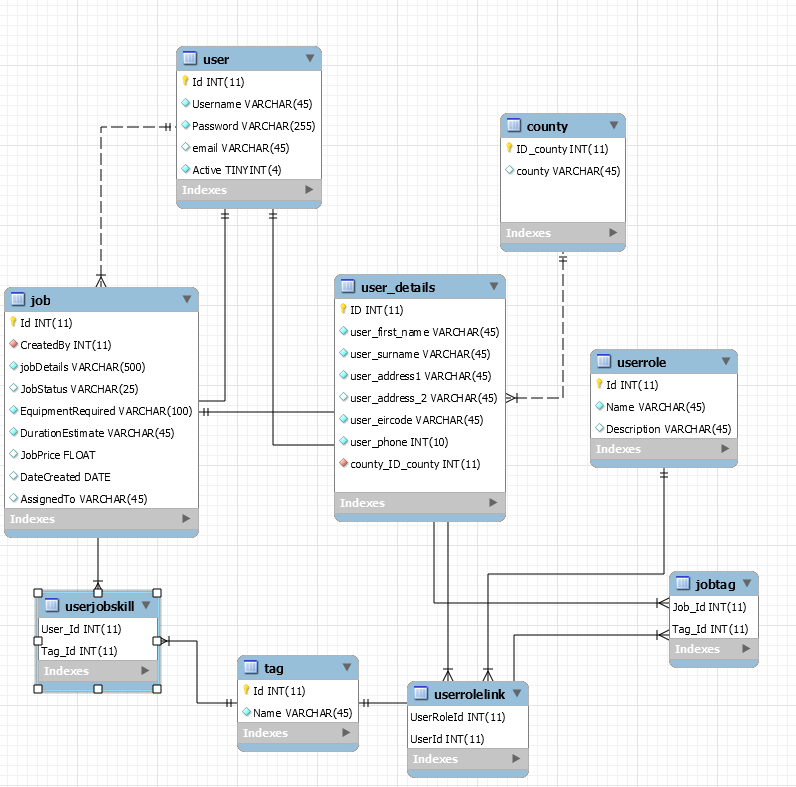


Figure EER Diagram Favours4Neighbours

### MVC

A model view control framework is used to aid in rapid application development for this web application. The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.

Model

The Model component is the data-related logic that the user works with. For example a Database or any other business logic-related data. For example, a Log In function will retrieve the user details from the database, validate the credentials and allow log in or deny entry to the system.

View

The View component is what the user sees and interacts with, the User Interface of the system. Depending on actions taken by the user the user uses the controller via the view which in turn updates the model to return the next view to the user.

Controller

Controllers act as an interface between Model and View components to process the logic and requests, manipulate data using the Model component and interact with the Views to display the corresponding view. For example, the Registration controller will handle all the interactions and inputs from the Registration View and update the database using the Registration Model adding a user to the database.

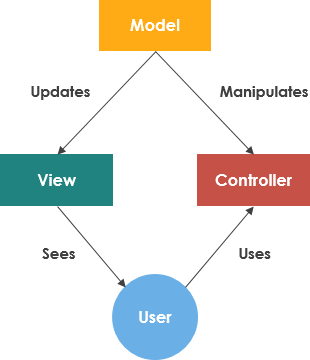


Figure Model View Controller Framework

### Logical Architecture Diagram

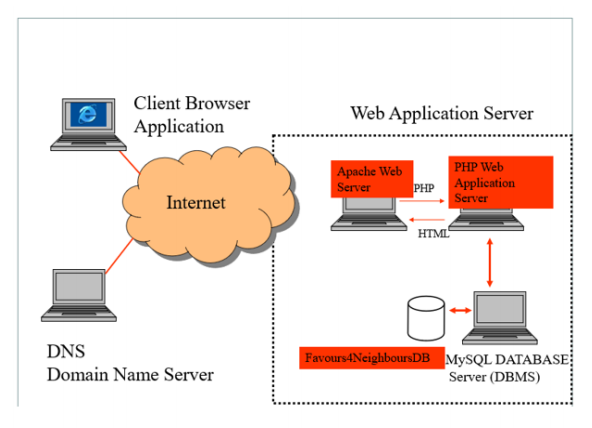


Figure Logical Architecture Diagram

## Implementation

### Introduction

### Burn down chart

### Screen shots of functionality

### Screen shots of database

### Class implementation code

## Conclusion

The development of this web application

Agile

Iterative process

Time

Security

## Bibliography

A. Singh, P. C. (2018). Formulating an MVC Framework for Web Development in JAVA. *2nd International Conference on Trends in Electronics and Informatics (ICOEI),* (pp. 926-929). Tirunelveli: 10.1109/ICOEI.2018.8553746.

ALONE. (2015, June 15). *ALONE launch Home First Campaign 2015 “Why are we forcing older people out of their homes?”.* Retrieved from alone: https://alone.ie/alone-launch-home-first-campaign-2015-why-are-we-forcing-older-people-out-of-their-homes/

BDO. (2014). *Health’s Ageing Crisis: Time For Action A Future Strategy for Ireland’s Long-Term Residential Care Sector.* BDO.

Health, D. o. (2020). *Overview of the Health System Response to date Long-term residential healthcare settings.* Department of Health .

Maven, A. (2021, January 15). *Welcome to Apache Maven*. Retrieved from Apache Maven Project: https://maven.apache.org/index.html

Nixon, R. (2012). *Learning PHP, MySQL, JavaScript and CSS .* O'Reily Media.

Nixon, R. (2012). *Learning PHP, MySQL, JavaScript, and CSS,* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Oropesa, R. (1993). Using the service economy to relieve the double burden. *Journal of Family Issues*, 7438-473.

S. Bianchi, M. M. (200). Is anyone doing the housework? Trends in the gender division of household labor. *Soc. Forces, 79*, 191-228.

Schmidt, D. (2013). *CIS501 lectures: Software Architecture.* Retrieved from http://people.cis.ksu.edu/~schmidt/501s13/Lectures/Lecture03S.html

Shah, N., & Balda Ortiz, G. J. (2013). *HTML5 Enterprise Application Development.* Birmingham, UK: Packt Publishing.

Standing, C. (2001). Methodologies for developing Web applications. *Informatio9n and Software Technology*, 151-159.

T. Lippe, K. T. (2004). Outsourcing of domestic tasks and time-saving effects. *J. Fam. Issues, 25*, 216-240.

Tzung-Jeng Hwang, K. R. (2020). Loneliness and social isolation during the COVID-19 pandemic. *Cambridge University Press Public Health Emergency Collection* , 1-4.

Welling, L., & Thomson, L. (2008). *PHP and MySQL Web Development* (4th ed.). Upper Saddle River, NJ: Addison-Wesley.

Wisniewski, J. (2011, 11 1). HTML5. *Online*, p. 4.