

# **IFB299 Application Design and Development**

**Semester 1, 2017**

**Jacob Bradford n9190007**

## **1. Introduction**

This document outlines my contributions to the project completed in IFB299 Application Design & Development. Primarily, my role within the project was a developer though frequently consulted the product owners and scrum master. In this document, it is clearly stipulated where my contributions positively affected the planning of the project and implementation of the requirements for sprints one and two. Evaluations of the work completed is also made and recommendations for improving the outcomes of future projects are stated. The appendices contain some sample artefacts which are complimented by the artefacts found in the directories with this document. Additionally, a detailed work log is in the appendices.

## **2. Contributions**

### **2.1. Planning**

During the planning phases of the project, I contributed towards the development of story cards; where acceptance criteria and stories were developed. This process involved me providing technical advice to the product owners. This advice supported the allocation of story points for each story card and some preliminary design discussions. This opportunity was used to ensure that the allocated time for the development team was appropriate. Estimations were made through negotiation, which resulted in an accurate story card.

Planning of the sprints required consultation with the story cards which had been developed prior. Again, I could provide advice on the order of implementation, ensuring that features could be constructed upon one-another. For example; discussions suggested that the implementation of the properties management system would be completed first but I suggested that this would not be realistic as it is dependent on other features, such as owner's management and staff management.

Preliminary discussions with the development team included suggestions on which technologies to use and methods of implementing the product. Initially, I suggested the PHP Laravel framework due to its diverse nature though further analysis of the project, it was revealed that it would be excessive for this use-case. Instead, it was decided that raw PHP code would be written as it is fully flexible and requires less setup than Laravel does. Furthering this, a Github repository was setup by another group member, also Jira.

I spoke with other developers to gain an understanding of their existing technical skillsets and they informed me that had prior experience with programming and some experience with the PHP language. Additionally, experience with client-side web technologies was apparent.

## 2.2. Sprint One

My contributions to the story cards in sprint one were significant. I built upon the existing work that Jason had started; the SQL scripts and the login & register forms.

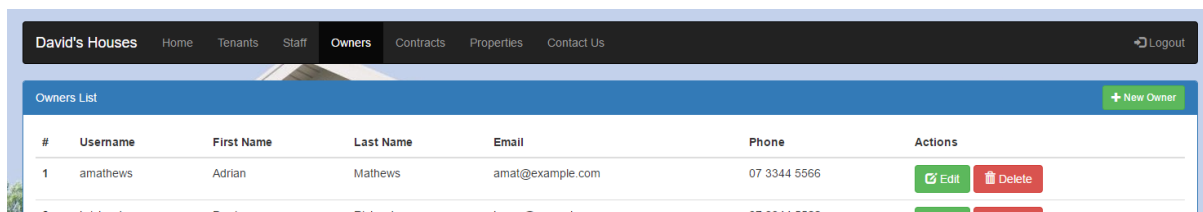
### 2.2.1. Artefact 1 – *Installation Script*

Through my prior experience in developing software, I knew that it is difficult to ensure all developers have an up-to-date database schema. This is due to the limited options of version control systems for databases. Instead, it is common practice to run scripts which migrate the database up and down; or move the schema backwards and forwards in versions. The implementation of a migration system for this project may be excessive. Instead, I developed an installation script which would create the schema and populate it with sample data. Each time this script was executed, the existing database would be dropped and a new instance would be created. In addition, my design warned the user if the database connection failed or if the schema was not setup properly. A sample of this script is available in the Appendices.

### 2.2.2. Artefact 2 – *Developed the create, edit, delete, list template*

As part of the initial setup of the product, I developed a template which lists instances of an object type (owners, staff, etc) and allows for creating, editing and deleting. This template can be enhanced for use throughout the entire system and was used modified on various occasions for use elsewhere (ie: the search results page).

A sample of this template can be seen in the *tenants.php* file in Appendices. A sample screenshot can be seen below.



#	Username	First Name	Last Name	Email	Phone	Actions
1	amathews	Adrian	Mathews	amat@example.com	07 3344 5566	<a href="#">Edit</a> <a href="#">Delete</a>
2	brichard	Borris	Richard	booor@example.com	07 3344 5566	<a href="#">Edit</a> <a href="#">Delete</a>

### 2.2.3. Artefact 3 – *Implemented basic search feature*

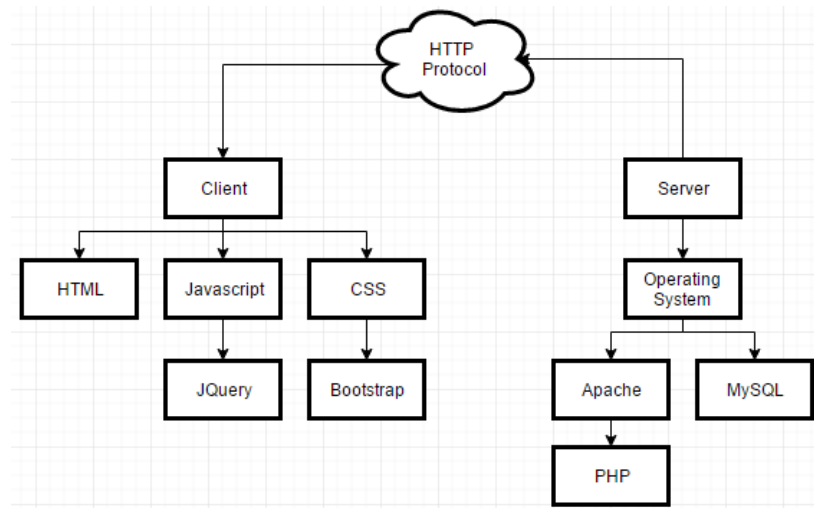
I implemented a basic search feature which was designed to find properties which matched a *like* query according to what the user entered.

A sample of the SQL query which does this is listed below.

```
1. SELECT *, staff.fName as staff_fname, staff.lname as staff_lname FROM properties
2. LEFT JOIN staff ON properties.staffID = staff.staffID
3. LEFT JOIN owners ON properties.ownerID = owners.ownerID
4. WHERE street LIKE '%{$_GET['q']}%' OR suburb LIKE '%{$_GET['q']}%'
```

#### 2.2.4. Artefact 4 – System Architecture

In preparation for the client presentation, I created the system architecture diagram, as seen in the diagram below. This diagram describes the technical system architecture of the product and would only be beneficial to developers or administrators working with the client. As a developer, I was primarily responsible for the *server* side of the diagram, whilst the end-user will only experience systems delivered on the *client* side of the diagram. By processing data at the server, clients, owners and the details of properties would be kept secure and private.



## 2.3. Sprint Two

My contributions to Sprint Two aligned those in the sprint plan and ordered by the scrum master. Most the work was to continue the implementation of the existing features and to finalise user permissions.

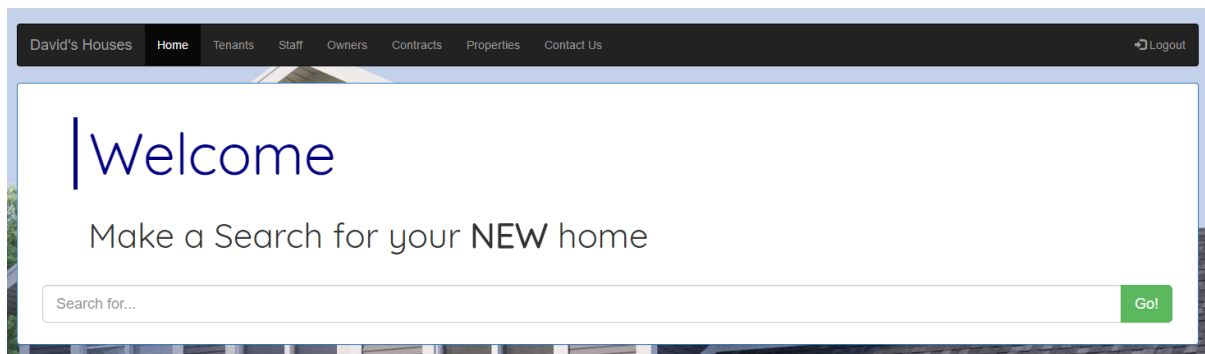
### 2.3.1. Artefact 5 – Suggested Developer for Task

The project was running accordingly to schedule but the time was remaining was limited. As I have the most experience with web development in the group, I devised a plan which stated who should complete each of the remaining story cards. This was a recommendation to the group and was not enforced. I assigned tasks based on time and complexity. The plan can be seen in the table below. Additionally, I added some notes on how I would implement the feature. This document was referred to by the group several times.

#	Name	Notes	Developer
1	Property ID reference	Property view page needs to built. It does already exists but (I think) it only lists the address right now. We should build a table that displays all the data (inc. inspection times?). Should this page only be viewable if it is not rented?	Michael
3	Viewing Times	Create a table something like this: <b>property_views</b> (id, property_id, start_datetime, end_datetime, staff_id). Create this table by adding to the install.php script. Then copy the contracts.php create/edit/delete method. It should be reasonably similar.	Jason
4	Property Inspection Registration	Create a table <b>property_view_tenants</b> (property_id, tenant_id). Again, copy the contracts.php file as it would be similar	Jason
7	Staff Property Allocation	Pretty sure this is done already?	
8	Property Assignment View	Copy the properties.php table which displays the list of properties in the system. Add a select box to the webpage and then modify the SQL query to do something like <b>WHERE staff_id=\$_GET['staffid']</b>	Michael
10	Separation of Duties	We'll do this last (ref: #16). Check for user_type and user_id in the \$_SESSION array to determine what they can see.	Jake
13	Website search refinement	On the home page, where the search feature is, add a few more textboxes which say 'suburb', 'street' and select boxes like 'number of bedrooms', etc. modify the SQL appropriately. ALSO, the properties create/edit/delete and database table needs to be updated to include number_of_(bedrooms bathrooms carports)	Michael
14	Property Photo Gallery	Create a database table using install.php; <b>property_images</b> (id, property_id, URL_to_image). Google search PHP file upload. Shouldn't be too difficult. Create a section in the property view page which finds all the images assigned to a property and display them. Also show a picture on the search results page?	Jason
15	Property Inspection Request	Create a new file called something like <b>contact.php</b> . Add a form which accepts an email, subject and message. When you click submit, make it print out 'We have received your email, etc'. But don't actually do anything with the email	Michael
16	User Login	Modify the existing login to allow the operator to select what type of user they are logging in as (staff, tenant, owner). If it david, his username is <b>david</b> then we know its david, regardless of what type of user the operator selects. Save the user_type and user_id in the \$_SESSION variable.	Jake
17	User Registration	same as login but registration, instead of login	Jake

### 2.3.2. Artefact 6 – Home Page Overhaul

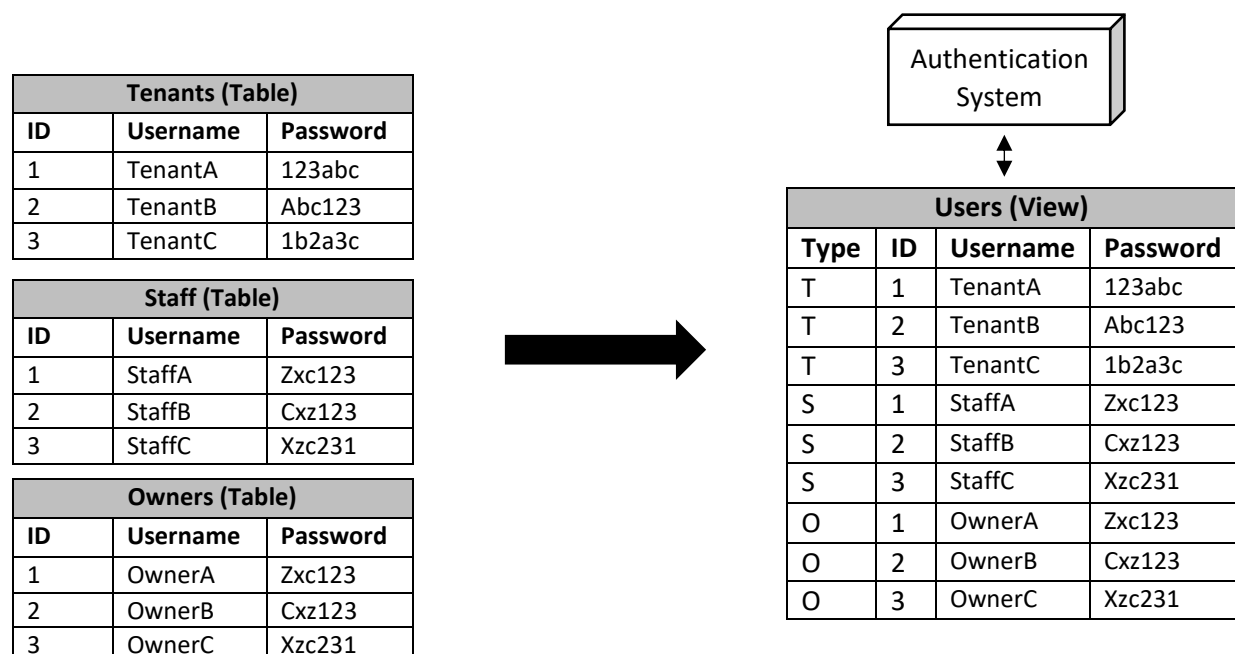
As the implementation of the product was nearing the finish, I made minor changes to the home page to make it look more appealing. This was in accordance to the feedback supplied by the client. As seen in the screenshot below, I have added some text and enlarged the search textbox. Further implementations and changes to the product could further redesign this page to make it more appealing. Eye-catching features such as a featured property or news updates could be added. This should be considered carefully as it serves as the primary point of contact for new visitors. It is recommended that in future implementations, a user interface / experience designer is appointed.



### 2.3.3. Artefact 7 – Login and Register

For the implementation of the login and register features, I used existing form designs and made modifications to the database schema. Notably, the registration form is a modified version of the 'New Tenant' page which staff members have access to. The login form was created independently from other existing features.

Owners, tenants and staff are stored in separate tables within the database, which results in the login system being required to check each of these tables for a matching user. Instead, I added a query to the installer script which creates a *view* of the tenants, owners and staff tables merged. A graphical representation of this can be seen below.



#### **2.3.4. Artefact 8 – *Separation of duties***

I implemented code to achieve the separation of duties between David, staff, owners and tenants. This restricted users to only seeing data that they had permission to view. The method which I implemented was static and requires changes to the code if permissions need to be changed. It is recommended in future implementations, the separation of duties is dynamic in such a way that an administrator can modify the permissions of each different type of user.

A sample of this work is difficult to supply due to its nature but is observable within the product by logging in as different user types. Notably, David has a username of *david*.

### **3. Conclusion**

In conclusion, the team performed well and developed a product for David to use. Notably, this product could be considered as a prototype version and future versions require further refinement, testing and stringency with the implementation of security methods. Additionally, code quality would be improved, whilst an appropriate framework would be used to ensure the ease of long-term maintenance.

Overall, I contributed significantly to the project, particularly to the technical implementation of the project. I implemented a develop-and-test approach where I would write code and complete functional tests on the feature.

Further work should include the use of a ticket management system and the full integration of version control, where branches and pull requests are used to ensure code is implemented without error or conflict. Additionally, a user experience and user interface designers should be integrated into the project team to ensure that the user-facing product meets industry standards and is appealing to the user.

## APPENDICES

### Work log

Date	Work Type	Description	Time Spent (m)
17/04/2017	Development	Modified some existing system architecture which Jason created. Added Bootstrap for web page styling, jQuery for client side scripting and started an installation file which sets up the database. This file is useful for when the application is pulled from the remote git.	30
17/04/2017	Development	Modified the search, login and register forms to use Bootstrap. Added a navigation bar.	30
17/04/2017	Development	Created the tenants management page. The page will list all clients in the system by default. The user can also delete a client from this list.	50
17/04/2017	Development	Created the ability to modify details of a tenant.	20
17/04/2017	Development	Added the ability to list, edit and delete contracts.	60
17/04/2017	Development	Added a <i>Create</i> button to tenants, staff and contracts.	5
17/04/2017	Development	Added the ability to list, edit and delete properties.	20
17/04/2017	Development	Added the ability to list, edit and delete owners.	5
17/04/2017	Development	Modified installation file to rely on a <i>config.php</i> file. This file has been added to the gitignore, allowing each developer to configure the database connection themselves, without their configuration being overwritten.	15
26/04/2017	Development	Added the ability to create tenants, staff, owners, contracts and properties. I reused the existing edit forms by writing some code to either select an existing item from the database or to use a fake temporary when creating a new item. Reworked some of the installation process and warning messages – it should be fool proof nearly.	30
26/04/2017	Development	Modified the existing search page which Jason created – it now connects with the database to show search results.	15
26/04/2017	Development	Added the error log to the gitignore file so that each developer has their own error log	5
26/04/2017	Development	Started the View Property feature to allow property profiles to be viewed. This was primarily needed so that the search page can link results to a more informative page. This page needs more work.	10
19/05/2017	Development	Added highlighting to the navigation bar as per the client request	10
19/05/2017	Development	Started implementing OOP style to allow for unit testing	20
19/05/2017	Development	Started implementing OOP style to allow for unit testing	120
24/05/2017	Development	Fixed property images issues	15
24/05/2017	Development	Made search bar & home page look better, removed OOP style code	40
25/05/2017	Development	Added separation of duties	30
25/05/2017	Development	Added ability to record username and password for tenants, owners and staff	10
25/05/2017	Development	Added ability for guests to register as a tenant	10



## Artefact 1

### Install Script Sample (install.php)

```
1.  <?php
2.  if(file_exists('config.php')) {
3.      include('config.php');
4.  } else {
5.      echo 'Please create the <b>config.php</b> file. Copy and paste the following into it.';
6.      echo '<br><br><span style="color:red">';
7.      echo '<?php $DB_SERVER = "localhost"; $DB_USER = "root"; $DB_PASS = ""; $DB_NAME = "ifb299_assignment"; ?>';
8.      echo '</span><br><br>';
9.      exit;
10. }
11. /*****
12.  * The intentions of this file is to allow for a quick
13.  * setup of the database. When modifying this file,
14.  * assume the database is empty.
15.  *
16.  * Always create at least 5 rows in each table!
17.  *
18.  *****/
19.
20.
21. // Connect to DB
22. $db = mysqli_connect($DB_SERVER, $DB_USER, $DB_PASS, $DB_NAME);
23. mysqli_query($db,"DROP SCHEMA ifb299_assignment");
24. mysqli_query($db,"CREATE SCHEMA ifb299_assignment");
25. $db = mysqli_connect($DB_SERVER, $DB_USER, $DB_PASS, $DB_NAME);
26.
27. /**** check connection ****/
28. if (mysqli_connect_errno()) {
29.     echo '<br><br><br>Failed to connect to DB<br><br>
30.     Ensure you have a database setup on <b>'. $DB_SERVER .'</b> and the user <b><?php echo $DB_USER; ?></b> has the password \
31.     '<b>'.($DB_PASS != '' ? $DB_PASS : '<i>blank</i>') .'</b>\'.<br>
32.     The database schema needs to be named <b>'. $DB_NAME. '</b><br>
33.     These settings are set in <b>config.php</b>. This file is ignored by git so it is unique to you.<br>';
34.     echo '<br><br>Use the script below to create the database and then run this installation script again.<br>';
35.     echo '<span style="color: red">CREATE SCHEMA "'. $DB_NAME. '"</span>';
36.     exit;
37. }
38.
39. /**** start installation ****/
40. mysqli_query($db, "START TRANSACTION");
41.
42. /**** add staff table ****/
43. mysqli_query($db,
44.     "CREATE TABLE IF NOT EXISTS Staff(
45.     staffID INT AUTO_INCREMENT,
46.     username varchar(40) NOT NULL,
47.     password varchar(40) NOT NULL,
48.     fName varchar(40) NOT NULL,
49.     lName varchar(40) NOT NULL,
50.     email varchar(60) NOT NULL,
51.     phone varchar(60) NOT NULL,
52.     PRIMARY KEY (staffID)
53. );"
54. ) or die(failed('createStaffTable'));
55.
56. /**** add staff ****/
57. mysqli_query($db, "
58.     INSERT INTO staff (username, password, fName, lName, email, phone) VALUES
59.     ('david', 'jonesb', 'David', 'Jones', 'bobby@example.com', '07 3344 5566'),
60.     ('dallen', 'allend', 'David', 'Allen', 'DavidAllen@example.com', '07 3344 5566'),
61.     ('gavies', 'dreg', 'Greg', 'Davies', 'Greg@example.com', '07 3344 5566'),
62.     ('bobjones', 'jonies', 'Bob', 'Jones', 'bobby@example.com', '07 3344 5566'),
63.     ('phildun', 'dundundun', 'Phil', 'Dunphy', 'phil@example.com', '07 3344 5566'),
64.     ('lukeee', 'phy', 'Luke', 'Dunphy', 'luke@example.com', '07 3344 5566');") or die(failed('createStaff'));
```

## Artefact 2

### List, Create, Edit & Delete Template

```
1.  <?php include('helper.php'); ?>
2.
3.  <?php
4.  /***** Create/Edit client *****/
5.  if (isset($_GET['save'])) {
6.      if (isset($_POST['id']) && $_POST['id'] > 0) {
7.          ...
8.      } else {
9.          ...
10.     }
11. }
12.
13. /***** Delete client *****/
14. if (isset($_GET['delete'])) {
15.     ...
16. }
17. ?>
18.
19. <?php include('pageHeader.php'); ?>
20.
21. <?php
22.
23. // List tenants
24. if (count($_GET) == 0) {
25.     $tenantList = mysqli_query($db, "SELECT * FROM ...");
26.     ?>
27.     <div class="panel panel-primary">
28.         <div class="panel-heading">
29.             Tenants List
30.             <span style="float:right; margin-top: -5.5px">
31.                 <a href='tenants.php?new' class='btn btn-success btn-sm'>
32.                     <span class='glyphicon glyphicon-plus'></span> New Tenant
33.                 </a>
34.             </span>
35.         </div>
36.         <div class="panel-body">
37.             <table class="table">
38.                 <thead>
39.                     <tr>
40.                         <th>#</th>
41.                         ...
42.                     </tr>
43.                 </thead>
44.                 <tbody>
45.                     <?php
46.                     while ($client = mysqli_fetch_assoc($tenantList)) {
47.                         echo "
48.                             <tr>
49.                                 <th scope='row'>{$client['tenantID']}</th>
50.                                 ...
51.                             ";
52.                         }
53.                     ?>
54.                 </tbody>
55.             </table>
56.         </div>
57.     </div>
58. <?php
59. }
60.
61.
62. // Create/Edit tenant
63. if (isset($_GET['edit']) && isset($_GET['id']) && $_GET['id'] > 0 ||
64.     isset($_GET['new'])) {
65.
66.     if (isset($_GET['edit'])) {
67.         // edit mode
68.         $tenant = mysqli_fetch_assoc(mysqli_query($db, "SELECT * FROM ... WHERE ...={$$_GET['id']}");
```

```

69.         $action = 'Edit';
70.     } else {
71.         // create mode
72.         // fill an empty array representing the new tenant. for quick hacks of the existing edit form
73.         $tenant = [
74.             'id' => '',
75.             ...
76.         ];
77.
78.         $action = 'New Tenant';
79.     }
80. ?>
81.     <form action="tenants.php?save" method="post" id="frmEditTenant">
82.         <?php if (isset($_GET['edit'])) { ?> <input type="hidden" name="id" value="<?php echo $_GET['id'] ?>"> <?php }; ?>
83.         <div class="panel panel-primary">
84.             <div class="panel-heading">
85.                 <?php echo "{$action} {$tenant['fName']} {$tenant['lName']}"; ?>
86.             </div>
87.             <div class="panel-body" style="padding: 2em">
88.                 <div class="form-group row">
89.                     <label for="example-text-input" class="col-2 col-form-label">Username</label>
90.                     <div class="col-10">
91.                         <input class="form-
control" type="text" value="<?php echo "{$tenant['username']}"; ?>" id="username" name="username">
92.                     </div>
93.                 </div>
94.                 ...
95.                 <div class="form-group row">
96.                     <div class="col-10">
97.                         <button type="button" id="save" class='btn btn-success'>
98.                             <span class='glyphicon glyphicon-ok-circle'></span> Save
99.                         </button>
100.                        <button type="button" id="cancel" class='btn btn-secondary'>
101.                            <span class='glyphicon glyphicon-remove-circle'></span> Cancel
102.                        </button>
103.                    </div>
104.                </div>
105.            </div>
106.        </div>
107.    </form>
108.
109.    <script>
110.        /* Client side form validation */
111.    </script>
112.    <?php
113. }
114. ?>
115.
116. <?php include('pageFooter.php'); ?>

```