|  |
| --- |
|  |
| IFB299 Application Design and Development |
| Personal Portfolio 2 |
| Name: Lucas Nielsen  Student Number: n8591733  Group Number: 21  Tutorial Session: Wednesday 2 – 4pm |
|  |
|  |

|  |
| --- |
| Five artefacts completed from user stories explained |

# Upload and View Multiple Images

In the *addproperty.php* and *addsHouse.php* webpages, I incorporated an html button that is capable of storing multiple images into an array for processing. From this point, the function ‘db\_getpropertyId’ was created in order to fetch the newly created property Id from the ‘v\_w\_house\_property\_details’ view in the database. This propertyId was utilised in the newly created function ‘db\_uploadnewimage’, which stored all the images into the ‘property\_pic’ table for viewing at a later date. Once uploaded, I was able to view the images through the modification of the *sHouseProperty.php* and *wHouseProperty.php* pages. Firstly, both these webpages incorporate a newly created function db\_getallimages which fetches the images from the ‘property\_pic’ table and is displayed on the page through the use a ‘for loop’ created by myself. The following files were chosen as part of my personal portfolio due to the fact that the upload multiple mage module is a ‘Must have’ feature and is crucial in the display of properties on the market.

# Carousel – Whole House and Share House

Similar to my first artefact, carousel involved the modification of the *sHouseProperty.php* and *wHouseproperty.php* pages in order to display multiple images as a slideshow feature. Firstly, the function ‘db\_getallimages’ (from the *db.php* file) was utilised to fetch all property images with the corresponding ‘propertyId’. This data was displayed through the use of a ‘for loop’ in pHp and CSS bootstrap styling. The primary reason for choosing these files as part of my personal portfolio is due to the fact that it utilised my view multiple image functions and enhances my skills in CSS bootstrap.

# Confirmation Email Registration and Password

In the *registerOwner.php* and *registertenant.php* pages, I managed to include the function ‘mail\_send\_registration\_code’ from the mail.php file which generated a confirmation email sent to the user with the incorporation of a verify email link. This function also utilised the ‘mail\_super\_switch’ function in the mail.php file that states the admin account credentials (i.e. my Gmail account). On the backend, the tenant or owner details are stored into a temporary table (tenant\_temp) on the database. Once the user clicks the link in the verification email sent, the user is redirected to the home page, however on the backend, the corresponding details stored in the ‘tenant\_temp’ table are transferred to either the owner or tenant tables on the database (depending on the completed registration form) through the use of the *confirmation.php* file. Both files mail.php and *confirmation.php* were created solely by myself through the enhancement of my previous created *contact-form.php* file in Iteration 1.

Continuing with the email functionality, I was able to modify the *updatePassword.php* file through the insertion of the ‘mail\_acknowledge\_password\_reset’ function from the *mail.php* file that sent an email to the user if the password to their account changed. Since these files are essential in the security aspect of the site and is majority a reflection of my work, I have subsequently included it as part of my five artefacts for this personal portfolio.

# Update ShareHouse

In the *updateSHouse.php* page, I created the html form that prefilled data based on the passed propertyId in the URL. When a user clicks the ‘edit’ button on the *sHouseproperty.php* page, they’re immediately redirected to the *updateShouse.php* page with the corresponding propertyId. This propertyId is utilised in the ‘db\_getSHouseDetails’ function to fetch property data from the ‘v\_s\_house\_property\_details’ view. All returned data is presented as a prefilled form, providing the user with the ability to update their property. When submitted, the form calls the ‘db\_update\_s\_house ‘ function, which utilises the ‘update\_s\_house’ stored procedure to update the property data. This file is majority my creation and is necessary in providing property owners with the ability to update their Share house if changes arise, hence it was included as one of my five artefacts for this personal portfolio.

# Add, Update and View Profile Picture – Share House and Whole House

In both the *registerOwner.php* and *registertenant.php* pages, I modified the html form to incorporate a field where the user can provide a profile picture. When the form submits, I also modified the ‘db\_addTenant\_temp’ function in the db file and the corresponding tenant\_temp table in order to account for the inclusion of the image field. When the user clicks on the verification email link sent and activates the *confirmation.php* file, I have modified the ‘db\_addOwner’ and ‘db\_addTenant’ functions and their corresponding owner and tenant tables to successfully upload the image to the database.

Once uploaded, the user can modify the profile picture through the *Ownerprofile.php* and *tenantProfile.php* files. A separate html button was incoportated in the form in order to update the image. When the form is submitted, the modified ‘db\_updateOwner’ and ‘db\_updateTenant’ functions were activated, which called the corresponding ‘update\_owner’ and ‘update\_tenant’ stored procedures.

For the user to view their profile picture, the *topNav.php* and *checkAccountType.php* pages were modified. The insertion of the ‘db\_getOwnerDetails’ and ‘db\_getTenantDetails’ functions in the *db.php* file provided the ability to fetch the profile picture and display it on the top navigation bar.