**Assignment 2: Technical Report**

By Andrew Parkhill (s5064284), Luke Powell (s5066140),

Daniel Pye (s4922402) and Michael Whatley (s5008606)

**Link and Credentials**

**System Design Rationale**

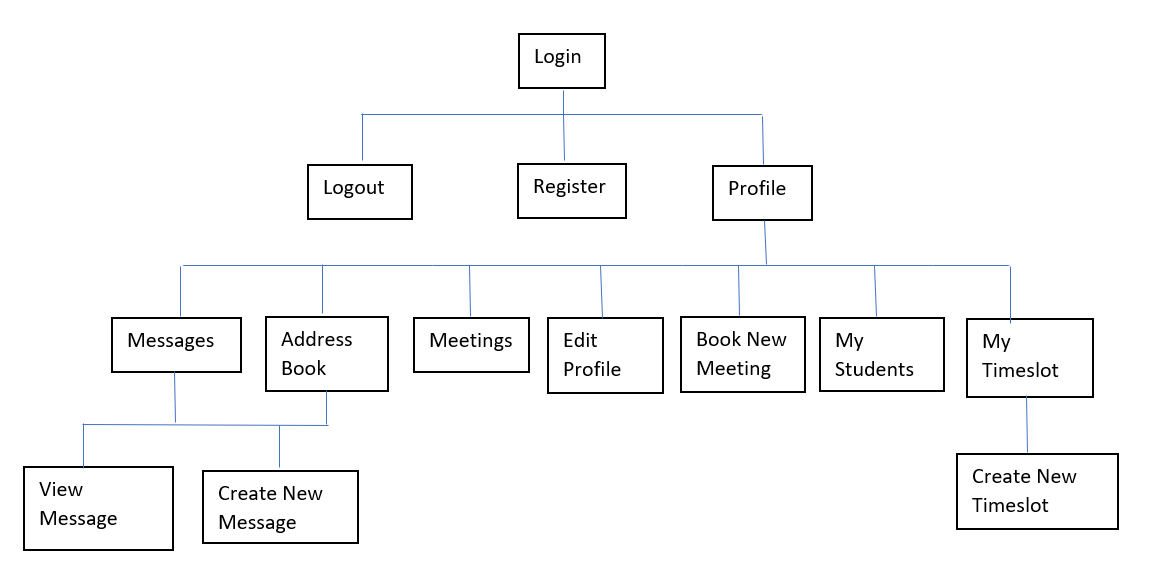
In considering the overall design of the application, it was agreed that we should aim for a web-based application style as opposed to the typical website layout. We agreed that the most professional layout would utilise a login page as the landing page which also allows us to ensure security from this point forward.

We also aimed for consistency throughout the application by implementing a navigation bar within the header which runs consistently throughout the application, adding and removing menu items dependent on the user’s individual access rights using sessions.

Forms and tables are used to display and edit information with clear buttons included to ensure it is convenient for users to reset fields if necessary.

**Navigation Design**

Detailed below is a navigation structure for the site:



**Database Design Rationale**

**User**

Information about all users of the system.

* **usr\_id** system-friendly unique identifier, auto-increments to ensure uniqueness.
* **usr\_email** user-friendly naturally unique identifier. Unlike usr\_id, we can display and expect users to remember this.
* **usr\_my\_key** stores the user’s password for authentication, hashed to improve security. This column is awkwardly named as an added security measure against unauthorised database access.
* **usr\_first\_name** stores the user’s forename for displaying.
* **usr\_last\_name** stores the user’s surname for displaying.
* **usr\_bio** stores user-defined self-descriptive text, nullable.
* **usr\_assigned\_lecturer\_id\*** points to the user’s assigned lecturer if they are a student, nullable.
* **usr\_auth\_level** stores the user’s role: 0 = student, 1 = lecturer, 2 = administrator, enables extra functionality for lecturers and administrators, defaultly 0 to improve security.
* **usr\_profile\_is\_private** stores whether or not students can view protected parts of the user’s profile, satisfies personal security needs, defaultly false to improve data availability.

**Message**

Information about private messages sent using the system.

* **msg\_id** system-friendly unique identifier, auto-increments to ensure uniqueness.
* **msg\_author** stores the author’s email address for clarification display purposes.
* **msg\_subject** stores the headline of the message for displaying.
* **msg\_recipient** stores the recipient’s email address for clarification display purposes.
* **msg\_date** stores the non-localised server side date and time that the message was sent, for displaying.
* **msg\_body** stores the content of the message for displaying.
* **msg\_read** stores whether or not the message has been read by its recipient, defaultly false to align with logic.

**Timeslot**

Information about times and places that lecturers have declared as available for meeting with their assigned students.

**tsl\_id** system-friendly unique identifier, auto-increments to ensure uniqueness.

**tsl\_start** stores the non-localised server side date and time that the timeslot begins, for displaying.

**tsl\_end** stores the non-localised server side date and time that the timeslot ends, for displaying.

**tsl\_lecturer\_id\*** points to the lecturer that owns this timeslot.

**tsl\_booked** stores whether or not a meeting has been arranged using this timeslot, defaultly 0 to align with logic.

**tsl\_location** stores the location where the timeslot takes place, for displaying.

**Meeting**

Information about meetings between students and their assigned lecturers.

* **met\_id** system-friendly unique identifier, auto-increments to ensure uniqueness.
* **met\_time\_slot\_id\*** points to the time slot used for this meeting.
* **met\_title** stores the headline for the meeting, for displaying.
* **met\_lecturer\_id\*** points to the meeting’s lecturer attendee.
* **met\_student\_id\*** points to the meeting’s student attendee.

**Security Considerations**

As a development team we took the decision to implement password hashing, which ensures a level of security from the outset. The password is hashed at registration stage, which ensures an encrypted version is held in the database and later used for validation.

Sessions have been implemented to further secure access rights. The authorisation level is assigned to a session array at login stage. This is then utilised throughout the application to ensure that the relevant information is displayed to the user i.e. ensuring lecturers and students only have access to the information they require

**MVC Framework/ Version Control**

As a group we made use of the CodeIgniter MVC framework which facilitated the separation of code in to a logical structure and allowed for a more logical approach for the group to follow. The separation of code between Models, Views and Controls allowed for the separation of concerns and allowed all group members to take responsibility for areas of the development and resulted in more concise files which are more readable for larger web applications.

The group also made use of GitHub for Version Control purposes which worked very successfully, allowing all members to work remotely and notify when any changes had been pushed to the repository.

**File Structure**

\*Turn this into an image/images to save words\*

Customised items in **bold**, Directories in black, Files in blue

**Assignment-2** (top-level directory, contains everything)

**Application** (contains the entire MVC architecture)

cache

**Config** (contains configuration information the system relies on)

**autoload.php**

**config.php**

constants.php

**database.php**

doctypes.php

foreign\_chars.php

hooks.php

index.html

memcached.php

migration.php

mimes.php

profiler.php

**routes.php**

smileys.php

user\_agents.php

**controllers** (contains controllers that are part of the MVC architecture)

**Meeting.php**

**Message.php**

**Timeslot.php**

**User.php**

index.html

core

helpers

hooks

language

libraries

logs

**Models** (contains models that are part of the MVC architecture)

**Meeting\_model.php**

**Message\_model.php**

**Timeslot\_model.php**

**User\_model.php**

index.html

third\_party

**views** (contains views that are part of the MVC architecture)

errors

**meeting**

**arrangemeeting.php**

**meetingsfooter.php**

**meetingsheader.php**

**meetingsitem.php**

**studentselection.php**

**timeslotselection.php**

**message**

**inbox.php**

**messagesent.php**

**newmessage.php**

**viewmessage.php**

**templates**

**footer.php**

**header.php**

**timeslot**

**addtimeslot.php**

**mytimeslots.php**

**user**

**addressbook.php**

**editprofile.php**

**login.php**

**mystudents.php**

**profile.php**

**register.php**

**registerSuccess.php**

index.html

.htaccess

index.html

**css** (contains Cascading Style Sheets (CSS) styling information)

**style.css** (contains styling information applied to the entire system)

**images** (contains image assets)

**administrator\_pic.png** (default profile picture for administrator users)

**lecturer\_pic.png** (default profile picture for lecturer users)

**student\_pic.png** (default profile picture for student users)

**sql\_scripts** (contains Structured Query Language (SQL) scripts for reference only)

**create\_meeting.sql** (creates the meeting table)

**create\_message.sql** (creates the message table)

**create\_timeslot.sql** (creates the timeslot table)

**create\_user.sql** (creates the user table)

**insert\_user.sql** (inserts sample data into the user table)

system (contains core CodeIgniter libraries and helpers etc.)

user\_guide (contains CoedIgniter documentation)

.editorconfig

.gitignore

**README.md**

**blank\_config.php** (template copy of config.php without environment-specific details)

**Blank\_database.php** (template copy of database.php without environment-specific details)

composer.json

contributing.md

Index.php (takes controllers and views as parameters to display the system)

license.txt

readme.rst

Reference: https://www.w3adda.com/codeigniter-tutorial/codeigniter-directory-structure

**[>1000 Words]**

**Appendix 1: System Testing**

**Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Purpose** | **Test Procedure** | **Expected Outcome** | **Actual Outcome** | **Pass/Fail** |
| **Registration** | | | | | |
| **1** | Ensure that registration functions and sends details to database | Enter all required fields in registration page | All details sent to database including a hashed password. Page redirects to profile page on successful registration | All details sent to database including a hashed password. Page redirects to profile page on successful registration | Pass |
| **2** | Email validation | Enter invalid email | Error message displays | Error message displays | Pass |
| **3** | Email database validation | Enter already registered email address at registration stage | Error message displays | Database error displays. Should implement a more specific error message | Fail |
| **4** | Checking required fields | Leave a field empty and try to submit form | Error message displays | Form resets without error message | Fail |
| **4** | **Re-Test** Checking required fields | Leave a field empty and try to submit form | Error message displays | Error message displays | Pass |
| **User Authentication** | | | | | |
| **5** | Ensure that login operates with correct credentials | Enter correct login credentials | Profile page for the user displays | Profile page for the user displays | Pass |
| **6** | Ensure that error message displays with incorrect credentials | Enter incorrect login credentials | Error message displays | Error message displays | Pass |
| **7** | Ensure that error message displays when required field not completed | Leave field empty and submit form | Error message displays | Page refreshes without error message | Fail |
| **7** | **Re-test** Ensure that error message displays when required field not completed | Leave field empty and submit form | Error message displays | Error message displays | Pass |
| **Profile** | | | | | |
| **8** | Check all the correct user credentials have been displayed | Ensure the loaded credentials match the correct record from the user table in the database | All the details loaded on the profile page match those of the logged in user | All the details loaded on the profile page match those of the logged in user | Pass |
| **9** | Check that correct links have been displayed in the navbar based on user’s authentication level | Review navbar | My Timeslots, My Announcements and My Students only visible for lecturers | My Timeslots, My Announcements and My Students only visible for lecturers | Pass |
| **10** | Ensure the ‘My Profile’ link loads the correct profile | Click ‘My Profile’ link | Link reloads the same profile | Link reloads the same profile | Pass |
| **11** | Ensure edit profile details save and implement as expected | Change profile Bio and set visibility as Lecturer only | Bio updates in profile and database and bio not visible to students | Bio updates in profile and database and bio not visible to students | Pass |
| **12** | Check the “clear” button in the edit profile page clears the bio field | Click the clear button when text has been added to the bio | The bio field clears | Only the additional added text clears from the bio | Fail? |
| **13** | Check that switching visibility to all users functions | Change visibility to “all users” | Bio visible in student profile | Bio visible in student profile | Pass |
| **Messages** | | | | | |
| **14** | Check that all messages for the user have been displayed in the messages table | Review data | All details from the message table display for the logged in user | All details from the message table display for the logged in user | Pass |
| **15** | Check that composing and sending a new message operates successfully | Click ‘New Message’  Complete all fields  Click Submit | Message displays to confirm that details have been successfully sent, and message table update with the new message | Message displays to confirm that details have been successfully sent, and message table update with the new message | Pass |
| **16** | Check that a composed message does not submit without required fields | Click new message and leave some fields empty before submitting | Error message displays | Form resets without error message | Fail |
| **16** | **Re-test** Check that a composed message does not submit without required fields | Click new message and leave some fields empty before submitting | Error message displays | Error message displays | Pass |
| **17** | Check how form responds to incorrect email address being entered | Enter incorrect email credentials and submit form | Error message displays confirming user not found | Error message displays confirming user not found | Pass |
| **18** | Check that clear button works on new message | Click clear button with text in fields | Text clears from form | Text clears from form | Pass |
| **19** | Check that delete message, deletes message successfully | Click delete button beside message | Message deletes from page and database | Message deletes from page and database | Pass |
| **20** | Check that view message functions | Click “view” beside required message | Message details display on separate page | Message details display on separate page | Pass |
| **Announcements** | | | | | |
| **21** | Check that announcement displays on student message page | Make announcement from lecturer profile | Announcement displays on student message page | Announcement displays on student message page | Pass |
| **22** | Check that delete announcement removes announcement from lecturer and student | Delete announcement while logged in as lecturer | Announcement removes from lecturer and student | Announcement removes from lecturer and student | Pass |
| **23** | Check that view announcement link functions | Click “view” link beside an announcement | All announcement details display | All announcement details display | Pass |
| **24** | Check that announcement form will not submit with empty fields | Click submit with empty fields | Error message should display | Form resets without error | Fail |
| **24** | **Re-test** Check that announcement form will not submit with empty fields | Click submit with empty fields | Error message displays | Error message displays | Pass |
| **Address Book** | | | | | |
| **25** | Check that search function searches for the details provided | Type a name in the search bar and click search | Returns all records containing that entered details | Returns all records containing that entered details | Pass |
| **26** | Check that search function only displays listed users | Type a name not listed in the Database in the search bar | No records should be returned | No records should be returned | Pass |
| **27** | Test message link beside a user in the address book | Click message link | Link takes you to new message form | Link takes you to new message form | Pass |
| **28** | Check that student cannot be added if they already have an assigned lecturer | Review address book list | Link to add student not available on those without assigned lecturer | Link to add student not available on those without assigned lecturer | Pass |
| **29** | When logged in as admin, ensure that delete user function operates | Click “delete” button beside user | User deleted from database | User deleted from database | Pass |
| **Meetings** | | | | | |
| **30** | Check that all scheduled meetings have been loaded from Database | Review data | All meetings involving the logged in user displayed | All meetings involving the logged in user displayed | Pass |
| **31** | Check the “new meeting” function | Click ‘New Meeting’ link | All scheduled time slots for the assigned lecturer displayed which allows the user to arrange a meeting | All scheduled time slots for the assigned lecturer displayed which allows the user to arrange a meeting | Pass |
| **32** | Check that form does not submit with empty fields | Leave fields empty and submit timeslot form | Error message should display | Form refreshes without error message | Fail |
| **32** | **Re-test** Check that form does not submit with empty fields | Leave fields empty and submit timeslot form | Error message displays | Error message displays | Pass |
| **33** | Check that user is unable to set two timeslots at the same time | Schedule two timeslots at the same time | Error message displays to state that timeslots cannot overlap | Form refreshes without displaying error message | Fail |
| **34** | Check ‘Select Timeslot’ link functions | Click ‘Select time slot’’ on the assigned lecturers list of available slots | Meeting title text field should appear. On submission, user should be redirected to their updated list of meetings | Incorrect error message when final timeslot booked  “Unfortunately, you cannot arrange a meeting because your assigned lecturer has no available timeslots.” | Fail |
| **34** | **Re-test** Check ‘Select Timeslot’ link functions | Click ‘Select time slot’’ on the assigned lecturers list of available slots | Meeting title text field appears. On submission, user should be redirected to their updated list of meetings | Meeting title text field appears. On submission, user should be redirected to their updated list of meetings | Pass |
| **35** | Ensure meeting cannot be scheduled when lecturer does not have an assigned student | Click “select timeslot” link as a lecture with no assigned student | Error message displays confirming that meeting cannot be scheduled | Error message displays confirming that meeting cannot be scheduled | Pass |
| **Logout** | | | | | |
| **36** | Check logout function | Click logout in the navigation bar and try navigate to profile via URL | Page logs out, returns to login page and destroys session (unable to navigate to profile without logging in) | Page logs out, returns to login page and destroys session (unable to navigate to profile without logging in) | Pass |

**Appending 2: Sample Data**

#### WP Assignment 2 – Group Self-Reflection Sheet

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Yes (completed)** | **Partially completed** | **No (not submitted)** |
| **Major requirements** |  | | |
| User registration | X |  |  |
| User authentication | X |  |  |
| Sending messages | X |  |  |
| Personal information /profile pages | X |  |  |
| Meeting arrangement | X |  |  |
| Listing/viewing | X |  |  |
| Searching | X |  |  |
| **System features** |  | | |
| MVC used | X |  |  |
| Sessions used | X |  |  |
| Input validation and sanitisation | X |  |  |
| Secure encryption via encoding or hashing | X |  |  |
| Additional features and extra functionality |  |  | X |

**List the third party libraries, frameworks, scripts, etc. that you used in your assignment:**

|  |
| --- |
| CodeIgniter |

**As a group, what is the best part of your work?**

|  |
| --- |
| The use of sessions to set pages dependant on authorisation levels |

**As a group, what is the weakest part of your work?**

|  |
| --- |
| We could have focussed more effort on the design phase in order to improve the usability of our final product |

**As a group, which part of the assignment you struggled most with?**

|  |
| --- |
| Differing programming styles made it difficult to maintain consistency in the code |

**As a group, have you experienced any challenges or issues?**

|  |
| --- |
| Generally a good experience. It was difficult to keep everyone to the same level, particularly in learning the use of the MVC framework |