

---

# FIT2104 Web Database Interface

## Assignment 3 Specifications

*Title: Web App Development Fundamentals*

---

### Notes

1. This assignment is worth 20% of the total mark for the unit
2. This is a **group** assignment, group repo will be available from week 3
3. See Moodle for deadline, extension and special consideration policy
4. **Complete the “Plagiarism Coversheet Electronic quiz” on Moodle before submit**
5. There are many details in this specification - make sure you read **ALL** of them

### Due dates

Preparation practices	By completing weekly lectures, quizzes and labs
Assignment checkpoints	During weekly labs and git commit records
Question & answer session(s)	- By posting your question on the discussion forum - By organising consultation with your tutor - Ask for help during weekly labs
Application Implementation	<b>Week 6 Friday 30<sup>th</sup> August @ 11:55 pm AEST</b>

### Scenario

#### Project descriptions

Nathan Jims is a professional recruiter specialising in B2B projects. He has been recruiting potential SMEs / contractors for business projects for a few years now. Until today most of the outreaching is being done by sending leaflets to local business chamber members and word of mouth via Nathan’s networking efforts. As the company is growing in size fast, Nathan is planning to build a website system to free him from the trivial administrative tasks so he can focus on outreaching and liaising with people, which is what he does the best.

■

You've been introduced to Nathan to implement the system with a limited, defined scope. This would help Nathan to determine if he can entrust you on the analysis, design and implementation of the full system in a later stage.

The expectation of this assignment is to develop a web-database application for Nathan that will enable the administration of client information, as well as a public-facing website to promote the aforementioned business projects that Nathan is recruiting SMEs / contractors for.

### General requirements

The database **must be a MySQL database** and the web application must use vanilla PHP (with pdo\_mysql interface). PHP rapid development frameworks (Laravel, CakePHP, CodeIgniter, etc.) are not allowed. The connection between the web pages and the database must be specified in **one place only** within your application (as a connection/configuration file or similar). Additionally, you must implement features which guard against SQL injection on all parts of the website (the minimum requirement in such regard would be the use of prepared statements throughout).

The requirements for this assignment are based on the learning objectives in FIT2104. Thus it's important that you will watch all lectures, read all required and extended materials, complete all assessed quizzes, as well as attend all labs and complete all lab exercises to date.

As the project has a defined scope, data model and most of the functions will be pre-defined for you in the format of a few resources listed on Moodle along with this assignment specification document. You'll need to implement the functionalities described in the following section, and provide **sufficient and realistic** demo data, and add them to the database to make the system usable when submitted:

- Sufficient data means each major entity (table) should contain at least 10 - 15 records, some records may contain uploaded files. To a degree that features in the system can be relatively well demonstrated
- Realistic data means you should not use random garbage (like "asdfasdf" as client's name) in your records. Instead use records that **look** legit and fit the purpose of **testing the system's functions**. This **does not** mean you should use real data (like real names for the people you know), and [a paragraph of placeholder text](#) can be used in longer fields. Images used **must be royalty-free**. Realistic data can be generated with services like [generatedata.com](#), and you can google for how to get royalty-free images

Although we admire your vision of having more features to help Nathan's business, **you should not implement those functionalities in this assignment**, as the scope of this assignment is defined. However you should consider discussing those functionality on the forum or during

weekly labs with your tutor, as it might provide help to enable you to achieve extra study goals, or implementing them on future assignments.

One last reminder - **ALL DATABASE ACCESS, FILE ACCESS AND CODE DISPLAY FOR THIS ASSIGNMENT MUST BE DONE USING PHP**

If you have any questions regarding the requirements of this assignment, you **MUST** discuss it with your tutors or the lecturer via Ed forum (preferred, so the answers are shared with everyone) and/or during the labs. Always get information from your tutor or the lecturer.

If not sure, just ask 😊


## Requirements

**Before we start**, please be reminded that your priority should focus on delivering functions and subfunctions in a whole - meaning instead of completing a function that is big, feature rich but doesn't work, deliver functions that are agile, precise and works. Start from MVP (minimum viable product), then add the nuance for better results.

**Database schema:** First, you will need to create a database with the given data model provided along with the assignment specifications. That means you'll need to decide if it contains all fields required to complete the following tasks, create a final schema for the project, whether the field name, data type, length and default value is suitable for each field, and the correct constraints for any foreign keys involved.

**Security/Authentication:** Security is a concern here and only Nathan (and potentially his staff) should have access to the system. A valid username and password must be entered in a login page to gain access to any of the pages on the site (except the login page and pages that are designed to be publicly available). Adequate security measures with passwords should be considered.

**Navigation and Dashboard:** A suitable way to navigate through the site should be considered. That means once Nathan has visited the website and logged in, he can go to all places by clicking links, instead of typing in file names in the address bar. Features should be accessible through some sort of navigation (sidebar, navigation bar, etc.) methods, and the user should be able see those navigation at all times.



Apart from navigation, the user should be on a dashboard after being logged in. You can either choose to leave this page simple or provide useful information, waiting for the user to go to other pages that carry concrete features.

In general, think about what a well-designed website in the public should flow.

**Projects pages:** The pages allow Nathan to browse and search projects. That means:

- A suitably formatted and sorted list/table which contains projects, **OR**
- A message indicating that no project has been found

And Nathan should be able to manage projects from a list. That means:

- Inspect the details of a selected project
- Add a new project, and assign the project to **one specific** SME / contractors
- Modify the details of the selected project and change the project's contractor
- Delete the selected project

In addition to above requirements, all user-input fields **need to be validated** to ensure data integrity. This can be done by either:

- HTML5 built-in form validation
- External Javascript validation libraries

And such validation should be implemented throughout the project in a consistent manner.

Also ensure that any database errors on these pages are captured properly, and a user-friendly error message is displayed to the user if any error occurs. Your application should not crash.


**Contractor pages:** This page will allow Nathan to manage (list, add, modify and delete) his clients.

Each contractor should have a profile picture to be uploaded. When the profile picture is being updated or the client is being removed, the old picture should be also removed from the file system. All uploaded client profile pictures must be stored in a subdirectory called **contractor\_profiles**

Also ensure that any database errors on these pages are captured properly, and a user-friendly error message is displayed to the user if any error occurs. Your application should not crash.

**Organisations pages:** This page will allow Nathan to manage (list, add, modify and delete) organisations of his clients.

Also ensure that any database errors on these pages are captured properly, and a user-friendly error message is displayed to the user if any error occurs. Your application should not crash.



**Users pages:** This page will allow Nathan to manage (list, add, modify and delete) users within the web database system, essentially who can log into the system.

Also ensure that any database errors on these pages are captured properly, and a user-friendly error message is displayed to the user if any error occurs. Your application should not crash.

**Contact Us page:** The contact page should include a basic contact form consisting of the name, email address, phone number and message the person who enquires would like to leave. Unlike most of the other features, the customer-facing form part of this feature should be made publicly available (without the need to login first).

Once the form is submitted, its content should be formatted into an easy-to-read format and sent to Nathan's email address: [nathan.recruiter@example.com](mailto:nathan.recruiter@example.com) (yes, this is not a real email address) using PHP's `mail()`.

While the data in the form is being sent via email, they should also be stored in the provided database table for records. There should also be a listing page for Nathan to manage those contact form fills, which allows Nathan to:

- Link a message to a certain contractor
- Track if a message has been replied to
- Delete unwanted messages

**Overall aesthetics:** You should implement Bootstrap templates to the website to make it more appealing to the users. Also you may implement various JavaScript plugins to further improve the usability of the website. This includes but not limited to:

- Use Bootstrap templates to make the website responsive (fits smaller screens)
  - Note this includes either you're designing your template with Bootstrap from the ground up, or use a **free** (as free-of-charge, you can **not** pay for anything out of pocket) existing Bootstrap template found online
- Use Datatables or similar to make tables more interactive and searchable
- Use Select2.js or similar to make dropdown lists/multiple select boxes more user friendly
- etc.

## Submission of the assignment

This assignment consists of several parts:

- The code implementation specified in this document

- Code quality (use of version control, meaningful commit messages, file organisation, code comments, code documentation, etc.)
- Development consistency (including conventions used throughout the codebase)
- Design thinking of features overall
- Professional conduct and practice (if applicable)

A detailed rubric/marking guide can be found on Moodle assignment portal.

For the code implementation, ~~you must create a repository within your FIT GitLab group repository with the name "fit2104\_assignment\_3" to submit the assignment.~~ The link to this repository will need to be submitted to Moodle's assignment 3 portal (see the portal for details). All HTML, PHP, SQL and any other associated files that make up the system should be in the repository's **main** branch (which must be the default branch). Apart from the predefined subdirectories (such as `contractor_profiles`), you can have more subdirectories within the folder to organise various files, such as `js/css/images` for static contents if required, and you can also organise PHP files by functions into directories. Your repository should not contain files that are not related to the assignment system. **Failure to keep the repository clean and tidy may result in deduction of the marks.**

At the root of your repository, you must include a file named `README.md` or `README.txt` (depending on the file format of your choice) that contains:

- Details of the author, with name, student ID#, and date of submission
- A link to your assignment's FIT GitLab repository page
- Credentials to access your system (username and password for login page)
- The filename of your schema and data SQL files (see below for details)

Create a SQL file (such as `schema_data.sql`) of your database schema and demo data. This file should include table creation statements with definition of primary and foreign keys and column constraints, as well as insert statements of all demo data. Which means the assignment marker should be able to use this file to create or restore the state of the database back to how it was at the time of submission.

Main page should be defined as **`index.html`** or **`index.php`**. This is to make sure when your assignment folder URL is accessed in the browser, such as this one:

`http://localhost/ass_3/`  
(Assume "ass\_3" is the folder which contains your system)

Your system will be displayed instead of a list of files or 404 errors.

A snapshot of your FIT GitLab group repository will be taken at the deadline of submission. By that time all files required for your system must have been **committed and pushed to your repository's main (default) branch along with the database files**. You will be able to make changes to your repository after the deadline, but those changes will not be considered for marking. **The git commit records should show the progression of the assignment at least every week by both team members.**

Consequently, please do not leave it to the last day or last minute to commit and push changes, as you'll need time to check if everything is properly submitted, and the server could crash because of the increased amount of traffic. No consideration will be given for failing to push changes before the deadline, and penalty will apply for late submission, unless exceptions or extensions are given according to the policies.

It's recommended to clone your final submission separately from your current working copy and test thoroughly if the code works as expected. Do **NOT** assume that just because the assignment works properly from your computer, it will also work as expected after being cloned elsewhere.

The authors must write all codes submitted. Students **may be** interviewed as a part of the assessment. Where there is evidence of plagiarism or collusion with those submissions, students involved will be subjected to academic integrity investigation according to the faculty policies.

Finally (something kind of out of the place but very important), any potentially dangerous act, such as attacking information systems, including maliciously crafted codes, or sabotage other students' assessments somehow, are considered as offences. Trust me, this happened before in the academic settings, and you don't want to go that far 🙄 I put it here just to cover all basis.

## Further Information

You are encouraged to use the Ed discussion forum on Moodle to post and answer questions regarding the assignment. Please make sure you have attached the correct tag for your post/question. It is important to remember that staff may not be available to answer questions at the last minute, so it is important that you do not leave it to the last minute to attempt the assignment.