## Criterion A: Planning

### The scenario

The client and adviser, [REDACTED], is the owner and CEO of [REDACTED] Pty. Ltd, which is a housing construction company that operates in Sydney, New South Wales (NSW), Australia.

In NSW, contractors working on a building site must have appropriate and up-to-date certification. Certification includes a valid and up-to-date Australian Business Number (ABN) and Service NSW licence. Otherwise, the company hiring the contractor has to pay the contractor’s business taxes. This has substantial implications on my client, as he would have to pay “tens of thousands of dollars in contractor’s taxes every year”.

Currently, there is no easy way to check whether a contractor has appropriately valid licence to work. The current system involves the client’s partner logging each contractor in a spreadsheet weekly and confirming contractor’s status’ weekly which is stored on a cloud drive account once confirmed. “It’s not ideal”, the client states, and if she didn’t have to do this task, she could allocate “time to do [...] productive things in the company.”

### Rationale for proposed solution

In “Interview 1”, we discussed possible solutions to use fewer human resources. The main goal we established was it should store and validate certification on a project-by-project basis. Supplementary goals we established were that the software should be easily maintainable, and it should be easy to use. Additionally, the software must be able to source data from a reliable position. Losing data would be a great detriment for operations and cause them to “receive [big fines]”. As the client already has a pre-established cloud storage workflow that works “well”, just the downloading of data is sufficient. This would allow the client to access their important information if there is a hardware failure/breakage. It should be available for use “on the go”- as they currently use mobile devices “when calling contractors and clients”. Also, a “to-do list” for further items of action to be noted is required.

In “Interview 2” we discussed the implementation details of a solution. Often the client is away from his computer and accessing/validifying contractor’s information “on the go” is ideal. A front-end website (developed in HTML, CSS & JavaScript) functional on both a phone and laptop would ensure this workflow is possible. The client could create a project on their main computer and save it to the cloud to download on their phone, etc. The use of Java in the backend will allow for the storage and sorting of data for project database management and searching. Java also allows for data collection using APIs, and for the importing and formatting of collected data to a JSON file.

### Success criteria

* Saves information for contractors per building project for multiple building projects
  + Client can get title or address, description, image of the project, future items to do about the project
  + Client can get appropriate information for contractors and projects
    - Name
    - Licence type
    - Licence number
    - Other appropriate pieces of information made publicly available about the contractors
* Search the list of projects to find the desired one
* Looks appropriate and functions well on mobile devices *and* desktop; performs well on either
* Be able to download data as a JSON file format for storage in already-established cloud storage system
* Read, create, and delete new contractors and projects

Words: 427

## Criterion B: Solution overview

### Record of tasks

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Planned action** | **Planned outcome** | **Time estimated** | **Target completion date** | **Criterion** | |
| 1 | Initial meeting with client, learning about situation and current system | Talk to the client about the project, learn about the requirements of the supervisor | 1 day | 06/09/19 | | A |
| 2 | Discussion with advisor about client, scope of project | Advisor approves the project in its current scope, and confirms my ability start prototyping | 1 day | 07/09/19 | | A |
| 3 | Prototyping user interface designs, researching underlying technologies | Start working through project requirements, start prototyping a browser-based UI that fits with requirements | 5 days | 12/09/19 | | A & B |
| 4 | Second meeting with client, finalising prototype designs | Talk with supervisor about the first prototypes of browser-based UI for further prototyping | 1 day | 14/09/19 | | A & B |
| 5 | Create software design based on MVC paradigm (UI, UML, DFD, and ER) | Design how the software components of the application will work together in according to agreed requirements | 1 week | 1/12/19 | | B & C |
| 6 | Creation of software components for the application | Begin the creation of the software components and commit to GitHub. | 2 days | 3/12/19 | | C |
| 7 | Development of the “View” software component | Work through developing the HTML, CSS, and JavaScript | 8 weeks | 15/02/19 | | C |
| 8 | Development of the “Controller” software component | Work through developing the backend in Java | 6 weeks | 01/05/20 | | C |
| 9 | Development of “Model” software component | Work through developing JSON file format | 1 week | 07/05/20 | | C |
| 10 | Field test with client in a project environment; implementation of solution | Test if the application is fit for purpose as per agreed requirements | 2 days | 10/05/20 | | D & E |
| 11 | Interview with client to confirm if developments are ok, and final adjustments to software components | Implement feedback from field test, adjust the software components accordingly | 5 days | 30/05/20 | | C & E |
| 12 | Creation of video with the project in use | Create a video to demonstrate how the application functions | 2 days | 05/06/20 | | D |
| 13 | Final interview and reflection with client, as well as personal reflection | Review the final functionality of the application, and how application can be extended | 1 day | 10/06/20 | | E |

### Design overview

**Model-View-Controller (MVC) diagram**

**A screenshot of a cell phone

Description automatically generated**

**Software Component “View”**

Figure 1: Initial Main UI



Figure 2: Initial In-depth Project UI

A screenshot of a cell phone

Description automatically generated

Figure 3: Initial Add New Project UI

A screenshot of a cell phone

Description automatically generated

Figure 4: Initial Mobile UI

A screenshot of a cell phone

Description automatically generated

Figure 5: Final Main UI

A picture containing computer

Description automatically generated

Figure 6: Final In-depth Project UI

A screenshot of a cell phone

Description automatically generated

Figure 7: Final Add New Project UI

A screenshot of a cell phone

Description automatically generated

Figure 8: Final Mobile UI

A screenshot of a cell phone

Description automatically generated

**Software Component “Controller”**

UML Diagram

A screenshot of text

Description automatically generated

**Software Component “Model”**

Entity Relationship Diagram (ER)

### A screenshot of a cell phone Description automatically generated

**Software Component Integration**

Data Flow Diagram (DFD)

### A close up of text on a white background Description automatically generated

### Function purposes

| **Function name** | **Function purpose** |
| --- | --- |
| + expandProject(projectNumber) | Use a designated project number in order to expand the front-end view of a building project |
| + search(link) | Search through list of all building projects to see if searched item appears in *any* places of a building project |
| + getLicenceID() | Retrieve the encapsulated licenceID variable |
| + getLicenceNumber() | Retrieve the encapsulated licenceNumber variable |
| + getBusinessNames() | Retrieve the encapsulated businessNames variable |
| + getCategories() | Retrieve the encapsulated categories variable |
| + getClasses() | Retrieve the encapsulated classes variable |
| + getLicenceName() | Retrieve the encapsulated licenceName variable |
| + getLicenceType() | Retrieve the encapsulated licenceType variable |
| + getlicensee() | Retrieve the encapsulated licensee variable |
| + getPostcode() | Retrieve the encapsulated postcode variable |
| + getStatus() | Retrieve the encapsulated status variable |
| + getSuburb() | Retrieve the encapsulated suburb variable |
| + getProjectID() | Retrieve the encapsulated projectID variable |
| + getClientName() | Retrieve the encapsulated clientName variable |
| + getStreetAddress() | Retrieve the encapsulated streetAddress variable |
| + getSuburb() | Retrieve the encapsulated suburb variable |
| + getDescription() | Retrieve the encapsulated description variable |
| + getContractorList() | Retrieve the encapsulated contractorList variable |
| + addContractor(Contractor contractor) | When given a Contractor object, add a contractor to an external database in JSON format |
| + getAllContractors() | Retrieve all Contractor objects stored on database in JSON format |
| + getContractorByID(String licenceID) | When given the licenceID of a Contractor object that has been stored, get the Contractor object and select it for below functions. If appropriate Contractor with licenceID searched has not been found, return nothing |
| + deleteContractorByID(String licenceID) | Given a Contractor object that has been previously selected using the above method, delete the object from database |
| + insertProject(Project project) | When given a Project object, add a contractor to a database in JSON format |
| + selectAllProjects() | Retrieve all Project objects stored on database in JSON format |
| + selectProjectByID(String projectID) | When given the licenceID of a Project object that has been stored, get the Project object and select it for below functions. If appropriate Contractor with licenceID searched has not been found, return nothing |
| + deleteProjectByID(String projectID) | Given a project object that has been previously selected using the above method, delete the object from database |

### Implementation plan to meet success criteria

|  |  |
| --- | --- |
| **Success criteria** | **Plan to implement** |
| Saves information for contractors per building project for multiple building projects  Client can get a title or address, description, image of the project, future items to do about the project | Implement a back-end data base in order to store appropriate data that the client wants to store (such as a project name, photos, and to-do list items), and can be accessed from the client-side |
| Search the list of projects to find the desired one | Create a search bar in a JavaScript front-end which can search through everything that is contained within a particular project (such as the names of contractors, addresses of particular houses, etc.) |
| Looks appropriate and functions well on mobile devices *and* desktop; performs well on both | Throughout development, continually use desktop and mobile versions of the website to ensure that usage is fine. Make sure application is somewhat lightweight and can handle weaker phone hardware |
| Be able to download data into a JSON file format for storage in already-established cloud storage system | Ensure that the database, and the client, have access to a method in which the database is downloaded into a JSON file through a button on the application |
| Have the functionality to read, create, and delete Projects | Develop REST API for Project using Java and Springboot, using their appropriate methods (GET, POST, and DELETE) |
| Have the functionality to read, create, and delete Contractors | Develop REST API for Contractors using Java and Springboot, using their appropriate methods (GET, POST, and DELETE) |

### Testing plan

| **Test scenario** | **Testing method** | **Expected result** |
| --- | --- | --- |
| Project view can be expanded | Click on project view when collapsed and expanded | Project view expands when already collapsed, and collapses when already expanded |
| Projects can be searched | Put through name of any certain project into search bar | Only the project with the searched project name shows up |
| Contractor can be added | Input sample contractor with random name, licenceID, and other points of information in JSON format | Contractor is inserted into database |
| Contractor can be retrieved from database | Input sample contractor licenceID | Contractor’s JSON data values are shown |
| Contractor can be added using their appropriate licenceID | Input sample contractor licenceID and all other appropriate fields | Contractor’s JSON data values are shown |
| Contractor that is selected can be deleted using their appropriate licenceID | Input sample contractor licenceID | Contractor with appropriate licenceID is deleted from database |
| Project can be retrieved from database | Input sample project projectID | Project’s JSON data values are shown |
| Project can be added using their appropriate projectID | Input sample project projectID and all other appropriate fields | Project’s JSON data values are shown |
| Project that is selected can be deleted using their appropriate projectID | Input sample project projectID | Project with appropriate projectID is deleted from database |

## Criterion C: Development

## A screenshot of a cell phone Description automatically generated

* JavaScript, HTML/CSS front-end (if/else statements + document.getElementById integration) function used to determine which button has been clicked on, and then whether it has to be expanded or contracted

A screenshot of a cell phone

Description automatically generated

* Search menu functionality to search through projects in View
* JavaScript front-end (if/else statements + JavaScript array) to determine if searched item (link) has been found in searchedList, is shown if searched and hidden is not searched (through CSS)



* Java back-end use of an abstract data type (ArrayList), stores a list of Contractor objects

A close up of text on a black surface

Description automatically generated

* Java back-end use of private final variables for Contractors in order to be encapsulated, such that unintentional access/change does not occur

A close up of text on a black background

Description automatically generated

* Java back-end getter methods for earlier private final variables for Contractors; setter variables not needed, as API always provides correct input data, and doesn’t need to change

A screenshot of text

Description automatically generated

* + Java, JSON, and SpringBoot back-end Object Oriented Programming use, in order to initialise the earlier private final variables, and declare the variables as JSON such that they can be sent to database for Contractors

A screenshot of a cell phone

Description automatically generated

Java + SpringBoot framework back end used in order to create REST API Controller for database connectivity, with appropriate POST, GET, and DELETE functionality to change database using JSON data for Contractors

* + POST ( addContractor() ) method sends a contractor to API Service in JSON
  + GET ( getAllContractors() ) method receives a list of all contractors in JSON
  + GET ( getContractorByID() ) method takes a given licenceID, and finds whether there is an associated Contractor JSON object in database
  + DELETE ( deleteContractorByID() ) method deletes any selected contractor, once given licenceID variable
  + *Similar methods were created for the “projects” class, with all appropriate private variables, encapsulation, constructors, JSON properties, and REST APIs developed*

A close up of a logo

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to GET information about Contractors from a locally established server
* Output is converted to a JavaScript object from JSON using JSON library

A picture containing drawing

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to GET information about Projects from a locally established server
* Output is converted to a JavaScript object from JSON using JSON library

A screenshot of a cell phone

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to POST user-inputted Contractor data to local API
* Converted from JavaScript object to JSON using JSON library

A screenshot of a cell phone

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to DELETE user-inputted Contractor data to local API
* Contractor to be deleted is the value defined in user input on website

A screenshot of a cell phone

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to POST user-inputted Project data to local API
* Converted from JavaScript object to JSON using JSON library

A picture containing drawing

Description automatically generated

* JavaScript used in front-end to create an XML HTTP Request to DELETE user-inputted Project data to local API
* Project to be deleted is the value defined in user input on website

A screenshot of a cell phone

Description automatically generated

* When using previous stored data, a file is inputted into addWrittenContractors / addWrittenProjects
* A GET XML HTTP Request is made to the file to get its recorded data, and then split into the splitWritten array by new line
* Once this is completed, the array is looped through and a POST request is made for all of the splitWritten items such that they are on the external server
* *A similar method to addWrittenProjects has been developed*

A screenshot of a cell phone

Description automatically generated

* To programmatically create new buttons for each project, a HTML “div” element is created and other elements are appended to it
* Elements are cloned from a template project (project0), and are stored in an array elementsToDuplicate, which is looped through
* Depending on what the id of the HTML element is, and the number of projects created, the program will change id to avoid duplicate HTML ids

## Criterion D: Flexibility and extensibility of product

## Criterion E: Evaluation

Upon speaking to the client in Interviews 4 & 5, it was “clear” that all of the previously outlined success criteria had been met, and the needs of the client were fulfilled. Already “being familiar” with websites allowed the client to read, create, and delete contractors and projects with ease. All of the data points for contractors and projects “were addressed”, and available to use. Similarly, the search bar was usable, and it took the client little time to get used to searching for projects. The output files “contractorData” and “projectData” worked “well enough” for the client, and the application worked well within the space and performance constraints of his phone. Within the success criteria, the client couldn’t find many faults in the application, and just suggested to “slightly tweak” the interface to make it more aesthetically pleasing, which would be ideal if he were to adopt the application for other people in his business.

One of the possible improvements brought up was the use of external, “government-created” APIs to get information about contractors. In manually adding data, there is a chance that provided contractor data “could be wrong”, which could cause issues with taxation and licensing. Additionally, this would also allow for the client to use the application in other states where APIs are available, as he doesn’t know contractors from outside NSW, which was a “possibility” for him in the future.

Another possible addition would be the “automated reminders” to a calendar or phone. The client already has a “online Google Calendar” solution and adding the functionality of being able to append a project to this solution could function to alert him more easily of when certain projects should be completed. This would use the Google Calendar API from the “Controller” section to send POST calls.

Additionally, another feature that was removed between Interviews 4 and 5 was to-do lists. Although the client had no need for the functionality, having to-do lists for each project would allow users to be more proactive, and not rely on other applications to perform the same functionality. To-do lists could also be responsive to the statuses of contractors in a project, and dynamically create to-do lists if the status of the contractor requires action (e.g. if their licence were to expire).

Words: 379 words

## Appendix

#### Interview 1

**Me:** Good afternoon, in this interview we should be able to establish a general idea of the problem and a solution that would fix it.

**Client:** Absolutely, good afternoon. Essentially, when working on a project, we hire a lot of contractors for every purpose, you know- tiling, electrical work, plumbing, and whatever else. Each one of those contractors has to have appropriate certification to be able to work, otherwise we get charged for the taxes they have to pay. There’s no way currently to easily be able to check whether a contractor has valid certification, and that’s the problem.

**Me:** What certification do you have to manage for the contractors? What is your current method of managing these contracts?

**Client:** The certification includes a NSW [New South Wales] Service licence for whatever job they’ve got to do. Currently, my wife inputs everything into a spreadsheet application after checking everything on a weekly basis. It’s not ideal, because if she forgets to check everything for one week when we’re on holidays or something we could receive a big fine – it would be like tens of thousands of dollars a year if at its worst. If she was free to not do this task, she could use that time to do actually productive things in the company.

**Me:** How do you use your current system? Would it have to be available to be used on the go, or would something like a stationary desktop application be more appropriate? How do you use your current system?

**Client:** I haven’t thought about that- we mainly use our current system on our laptops at home and bring them out to sites when we have to meet a new contractor. It would be quite useful to have a solution on the go, on our phones. We currently use our phones a great deal when calling contractors and clients for our company, so a way to use the solution on our mobile devices would be nice.

**Me:** A mobile system sounds doable and a nice solution to your problem. I imagine that you would also want it to run on your computer at home as well?

**Client:** That would be nice so that we could rely on both platforms, and it wouldn’t take that much time to learn over our current system.

**Me:** Do you have any other concerns with the operation of the solution? Anything else that you want me to consider when developing it?

**Client:** Yeah, a few things. I think it would have to be easy to use for people of our age, and something that we could learn to use in a short amount of time. We’re constantly getting new clients and building more houses, so the less time it takes to learn, the more time we can save and devote to building better houses. Also, it has to be secure and not lose its data or be compromised or something like that. One benefit of our current system is that it’s backed up on the cloud, and secure. We always know that it’s going to be there and that we’re not going to lose our information.

**Me:** Absolutely, I imagine that you might want a similar cloud-based system with the new solution as well. Anything else to add to the solution?

**Client:** ...Oh- like a small to-do list for each project so that we have actionable points to plan out our projects and any further action to do with contractors. And, yeah, I think we’re good.

**Me:** A to-do list sounds like a good idea. Thanks for your time, we will schedule a meeting after I’ve made some concept designs and mock-ups.

**Client:** Thanks, until next time.

#### Interview 2

**Me:** Thanks for making time again. I have come back with mock diagrams using all the information you gave me last interview, and here they are.

\*Refer to Figure 1 in Mock Diagrams – Appendix\*

**Me:** Would something like this be appropriate for the main view on a laptop?

**Client:** Yeah, this looks nice. I think it contains everything that we would need to use. Is this just the only screen? Or does something happen if you click on one of the project boxes?

**Me:** So this is just the main screen that’s used to give you information about everything that’s stored on the website, and then you will be able to click on each of the projects to get a better view of them and edit whatever you need to.

**Client:** Brilliant. If we’re talking about looking at everything from a glance, would it be possible to get a box up the top or somewhere that rounds up all possible problems in each of the projects? Like if a contractor was to have an item of certification expire soon?

**Me:** Yes, that would be a great idea. I will put that in the final mock-up. I could probably implement upcoming to-do items as well if you want those throughout the project too- I believe I did not add those in the mock-ups.

**Client:** That would be great, cheers.

\*Refer to changes in blue in Figure 5 in Mock Diagrams - Appendix\*

**Me:** Onto the next, this is the screen that you will get after clicking on any particular project.

\*Refer to Figure 2 in Mock Diagrams - Appendix\*

**Me:** This is what it would look like when you click on a specific project. If you want me to add time stamps for each to-do item that would be possible in the to-do items box.

**Client:** Yep, that would be good. Also, we need to go through what information to display in the contractor’s box. Do you know what you can find online and where to find it?

**Me:** Yes, I had a look through all of the websites and APIs that could lend data, and I think that getting the expiry date of their NSW Services licence and ABN. What else would be required?

**Client:** Something like the role or job that they’re registered to do, because if they do something else that they don’t have a proper license for the same tax issues come into play.

**Me:** Oh, ok. I will make a note of all of those for the final mock-ups. Anything else on this page?

**Client:** Nothing else, onto the next one.

\*Refer to changes in blue in Figure 6 in Mock Diagrams - Appendix\*

\*Refer to Figure 3 in Mock Diagrams - Appendix\*

**Me:** This is the screen that you would get when you add a new project from the main screen that we looked at earlier. As you said, the dates have to be added to the to-do list section. Anything else you can see that you would want?

**Client:** Again, with the contractor’s section, could we be a little more specific? Like, could you get it to give me a list of everyone with a specific name, Service NSW or ABN? Other than that, and the date added thing, it looks good.

**Me:** I do not think that we would easily be able to get API calls on the go for the ABN because of the way the government has structured the API, but I think that should be fine to do with the Service NSW and names.

**Client:** Ok, that’s alright.

\*Refer to changes in blue in Figure 7 in Mock Diagrams - Appendix\*

**Me:** The final mock-up is just of what I think it would look on your phone if you wanted to go mobile. There’s no real difference between this and the main view, just optimised to look better on a smaller screen.

**Client:** Yeah that looks fine. I imagine that the rest of the functionality will do the same thing, like adding new projects and all that?

**Me:** Yes, that’s the aim.

**Client:** Too easy.

**Me:** As for the logistics with how to develop it, I’m planning on developing a website. That way it will be accessible to you both on your laptop and on your phone easily, and I have previous experience developing- so debugging and development will take less time.

**Client:** What about cyber-attacks and other things with compromising data? As I think I said in the first interview, if we lose our data then we’re screwed.

**Me:** I was thinking about that. I’ll try my best to make the website as secure as possible, and I have had previous experience working on websites so that shouldn’t be too hard to cover. There are backup methods of security that we will have to use, however. Can you think of a way that you would use to do that?

**Client:** ... Could we download the data or something so that we could store it in the cloud?

**Me:** Yes, something like implementing a download as spreadsheet system could be implemented. Would you prefer to use that over a system inbuilt into the app?

**Client:** We’ve got our current system that works well managing cloud storage of data, so I don’t think that we need an implementation of it directly in the app. Other than that, ...I’ve got nothing.

**Me:** Alright, thanks for that. We’ll schedule another interview when I’ve finished development.

**Client:** Alright, cheers.

#### Interview 4

**Me:** Hi there, thanks for being patient and waiting for the final product. Here’s the product and how it functions.

\*Showed the functionality of the product by running it on laptop\*

**Client:** Oh, that’s pretty cool. I was now just thinking about it, and I probably wouldn’t need photos. You know, when constructing a new project it’s nothing more than a whole in the ground, so it probably isn’t necessary. And now that you mention it, we just bought a to-do list application last week so we probably wouldn’t need that feature in there anyways.

**Me:** Oh, ok. I can remove those things... that should be no worries. The data is all downloaded to a central file – those being called contractorData.txt and projectData.txt, and the project would function when used in mobile.

\*Demonstration of those two things\*

**Client:** Oh, that’s good. Could we sample some of the information that I would put in for one of the projects I’m doing?

**Me:** Sure.

\*Proceeded to complete successfully and end interview\*

#### Interview 5

**Me:** Hi again, thanks for being here for the final interview. I’m going to ask you a series of questions about the application and what you think of it. If I could also ask, please be as honest as possible when answering.

**Client:** That’s fine.

**Me:** Do you think that the functionality of the application matched the success criteria we established earlier, and your general use cases?

**Client:**

**Me:** Do you think that the application is genuinely better than the system that you use now? And if so, how?

**Client:**

**Me:** Do you think that, within the success criteria, any part of the project could be improved? If so, what, and how?

**Client:**

**Me:** Outside of the success criteria we established a while ago, what do you think could be improved about the project?

**Client:**