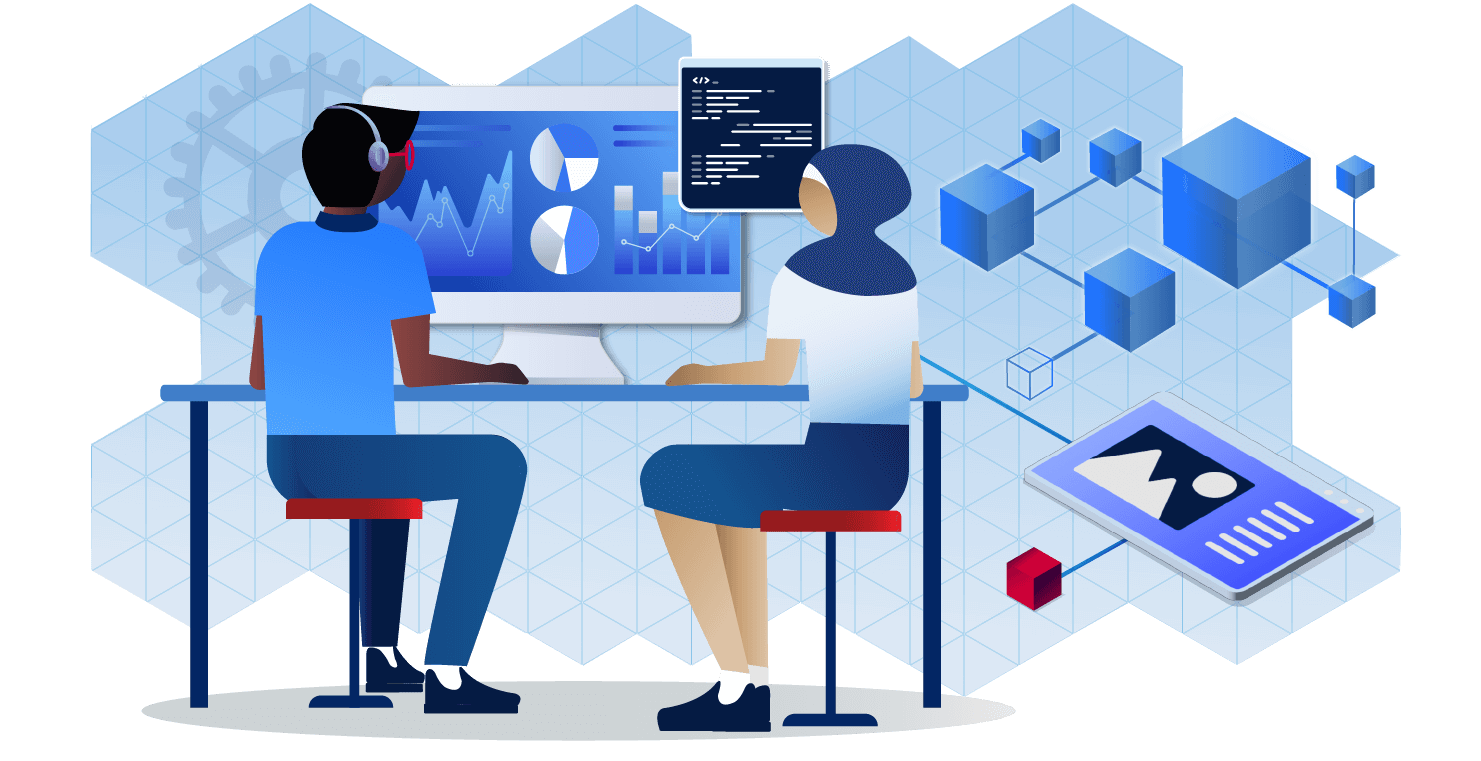
# Software Engineering Stage 6 (Year 12) – Sample Software Engineering Systems Report Template



Contents

[1. Identifying and defining 2](#_Toc135061669)

[1.1. Define and analyse problem requirements 2](#_Toc135061670)

[1.2. Tools to develop ideas and generate solutions 4](#_Toc135061671)

[2. Research and planning 8](#_Toc135061672)

[2.1. Project management 8](#_Toc135061673)

[2.2. Quality assurance 10](#_Toc135061674)

[2.3. Systems modelling 13](#_Toc135061675)

[3. Producing and implementing 19](#_Toc135061676)

[4. Testing and evaluating 20](#_Toc135061677)

[4.1. Evaluation of code 20](#_Toc135061678)

[4.2. Evaluation of solution 21](#_Toc135061679)

## 1. Identifying and defining

### 1.1. Define and analyse problem requirements

**Problem context**  
Students **analyse** the problem by **describing** each of its individual components and **explaining** how each of these components contribute to the problem needing resolution.

Hanh Le is a Chinese acupuncturist located in Sydney Australia. She has her own practise, however her current website serves as a major obstacle to potential exposure and clientele. The website is a free website made with WIX and thus features a large watermark. Additionally, the website does not provide much information about her or her services and does not have a booking option. Finally, the page isn’t at all aesthetic, the user experience is poor as the fonts size and colour do not match the background. Thus, the issues with her website warrant resolution.

**Needs and opportunities**Students **describe** the needs of the new system to be built based on the problem context and using the table given below.

|  |  |
| --- | --- |
| Need | Description |
| 1. A clean and effective UI | The user interface will need to have a consistent appearance, accessibility features for an effective user experience which can come in the form of the internationalisation, as the client base is predominantly non-native English speakers. Furthermore, intuitive navigation will be implemented to ensure user retention. |
| 2. About me page | What was missing from the previous website was a description of the practitioner or the services. An about me page will be implemented to provide users with insight into the practitioner and a brief description of what is being offered to orientate the user to the website, along with contact information and the location of the clinic. The about me page will be configurable by the Practitioner. |
| 3. Schedule/Booking | An online schedule would allow individuals to see  when the practitioner is available and allow users to book into a slot. This mitigates users having to manually contact the practitioner which can be tiresome and nerve-wracking, thus increasing potential revenue. The schedule would also allow the practitioner to view the bookings and set a schedule. |
| 4. Blog Page | To increase user retention and educate viewers, a common characteristic of professional practioners websites is a blog page which displays author posts and subsequent discussions. For this I will use the Instagram API to connect to recent posts which allows for engagement with the Instagram page and allow the practitioner to write informative posts. |
| 5. Admin Dashboard | The previous website did not allow the Practitioner to change portions of the website with ease. Thus, an Admin Dashboard will allow the Practioner to modify the blog page, about me page and add customer testimonials, where changes can be made and loaded in Realtime. The Admin Dashboard must be intuitive and perceivable given the clients lack of programming proficiency. |

**Boundaries**

Students **analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc*.*

Boundaries:

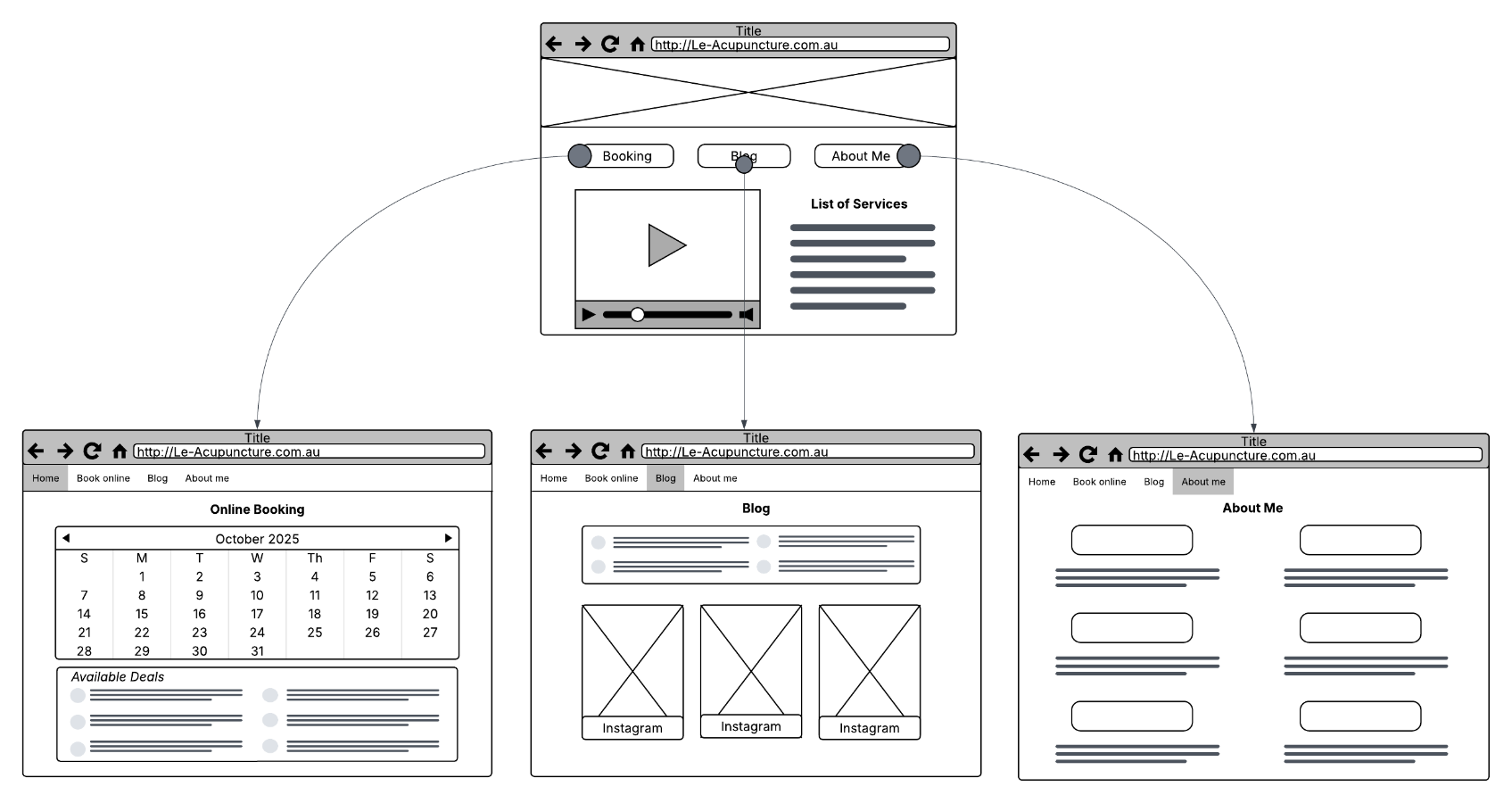
1. Scheduling (API)
2. Security
3. Server deployment

The limitations of the new system will be contrived by my lack of programming expertise, pertaining to the scheduling, security and server deployment components. The scheduling functionality of the website will be implemented using an API from an external source. This is because I do not have the expertise to build my own scheduling software where users are able to observe and book into a schedule with automated messaging and admin controls, appropriate for a real business. My expertise only extends to simple attacks such as CSRF, XSS, SQL attacks, however if concerns arise phishing attacks are targeted toward the practitioner who is not technologically savvy, or denial of service attacks. The server deployment is boundary, because at the moment I lack the expertise in setting up a home server or using cloud servers to host the website.

### 1.2. Tools to develop ideas and generate solutions

**Application of appropriate software development tools**Students **apply** appropriate tools such as brainstorms, mind-maps, storyboards and prototypes.



****

1.3. Implementation method  
Students **explain** the applicability of the implementation method for the current project. These methods are normally direct, phased, parallel, or pilot.

The implementation for the current project will be a pilot implementation. I have chosen this method, which is essentially a proof of concept, over more direct methods, is to test the functionality in a limited scope, identify potential issues before the sever deployment. The pilot implementation will consist of gathering a small group of users to use the site as normal and test functionality to identify bugs, issues or room for optimisations. The decision to choose a pilot implementation was made because of the scheduling portion of the website, as if directly deployed and the scheduling method didn’t work it could lead the potential loss of customers, or errors in booking. Thus a pilot program would allow for the feedback, the identification of issues before the full-scale deployment, further allowing for evaluations and improvements.

### 1.4. Financial feasibility

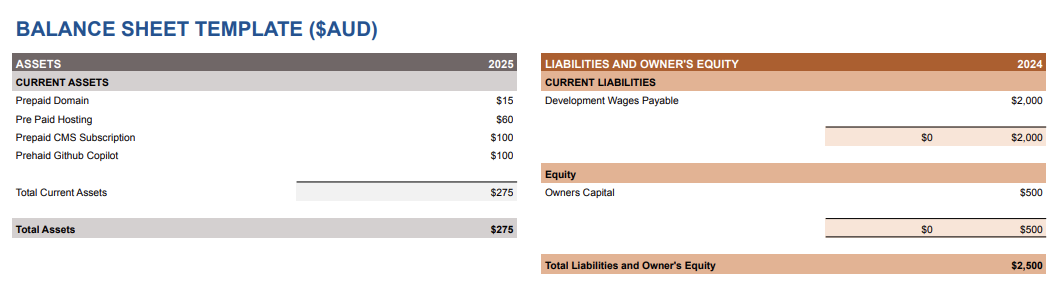
Students are to **conduct** a financial feasibility study, including producing an opening-day balance sheet, to assess whether their application is financially viable.

##### SWOT Analysis

|  |  |
| --- | --- |
| Strengths   * Developer experience in creating Progressive web applications * Strong relationship with client * Client is already established in the community * Regular posts and blog post builds SEO | Weaknesses   * My time management skills * Lack of proficiency in website deployment * Low initial traffic and visibility (will rely on paid promotion, costly) |
| Opportunities   * The use of external API’s to increase website functionality * SEO and Niche targeting * Use of analytics software for insights in how to gain customers | Threats   * Client can have a hard time articulating wants and needs * Security breaches * Competition against competing practitioners with established SEOs |

|  |  |  |
| --- | --- | --- |
| Study | Go or No Go? | Assessment and evidence |
| Market feasibility | No Go | The product is not market feasible as it is a personalised website for a small acupuncture clinic. The only way it could be market feasible if it became a template for small business to use which is unfeasible as the CRM market is extremely competitive. |
| Cost of development | Go | Based off the balance sheet, the theoretical cost of development is $2000, as typical freelance builds start at $1500 - $3000 for basic feature sets. However, as the developer the project will be undertaken as pro bono work. Meaning there is minimal risk based off the non-existent developer costs. |
| Cost of ownership | Go | As calculated based off the balance sheet total cost of ownership based off hosting, domain and CRM costs are ~$275 per year, which based off conversing with client is negligible. |
| Income potential | Go | The largest contributor to lost income for small practises is no shows according to Module MD. The scheduling/booking systems that provide automatic reminders are shown to create a 34% reduction in non-attendance according to Dialog Health, allowing for increased income potential. |
| Future expansion opportunities | Go | Future expansion opportunities include an e-commerce addition to the website, allowing the practioner to sell branded herbal products which can be advertised online and in person to pre-existing customers within sessions. The Global Herbal Supplements is projected to grow from USD $92 billion to $204 billion in 2032. |

##### Opening Day Balance Sheet



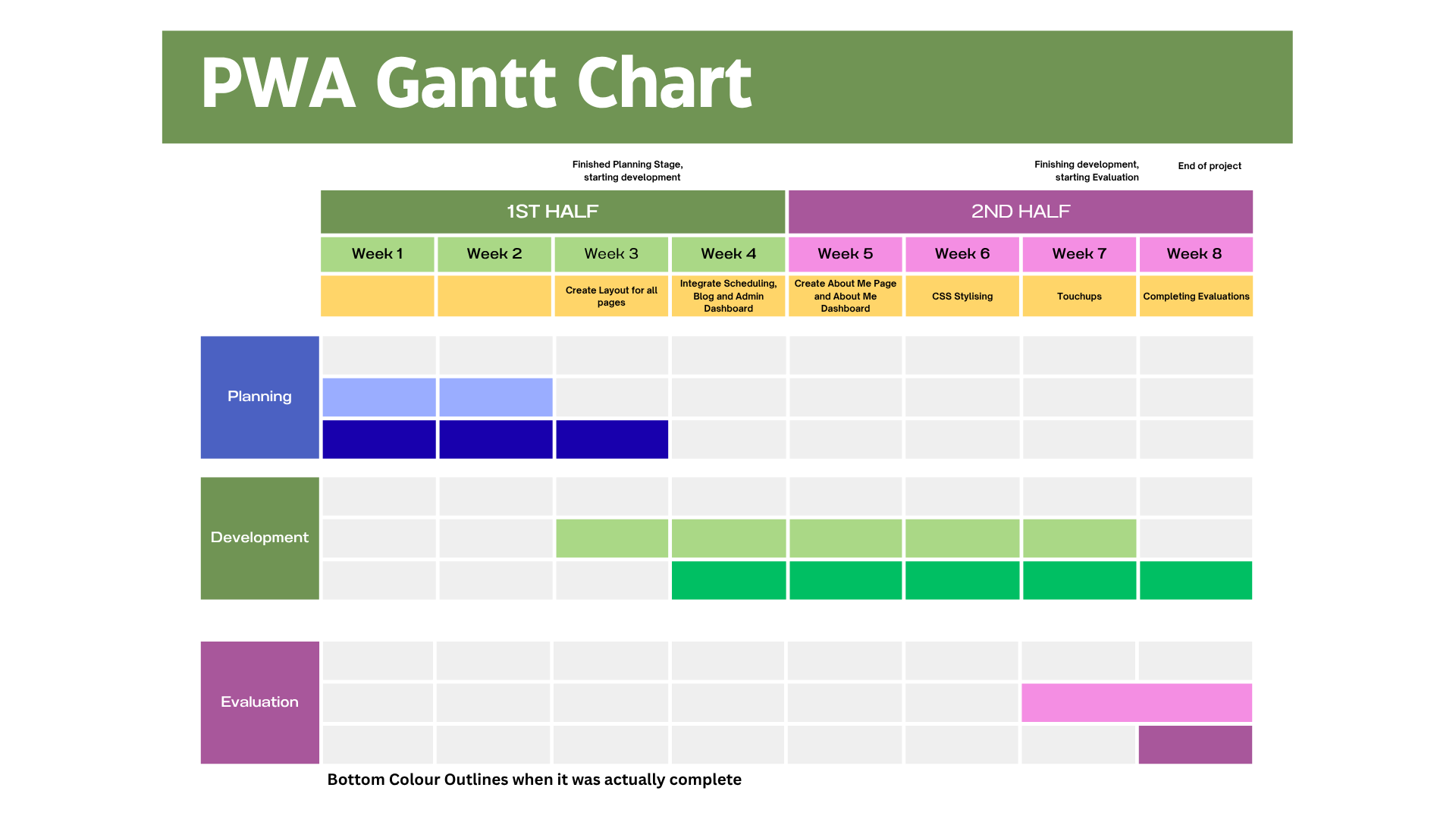
## 2. Research and planning

### 2.1. Project management

**Software development approach**Students **explain** the software development approach most applicable for this current project. These are normally: Waterfall, Agile and WAgile.

Agile is the most fitting software development approach as it provides incremental delivery, flexibility for evolving requirements and a continuous feedback loop. The phased delivery allows for the layering of functionality, allowing the practioner to interact with modules from the first sprint. This is specifically important, as the practioner could idealise new requirements emerging through live system usage, for e.g. she decides she wants to implement telehealth, which can be addressed in the next sprint. Agile permits these changes rather than being locked into rigid specifications. Furthermore, the sprint cycles allow regular reviews of progress with the practioner, who is not the most articulate, so the hands-on feedback on usability, branding etc allows for the product to match the practitioners expectations.

**Scheduling and task allocation**  
Students **develop** a Gantt Chart that details the tasks required to be completed, person or people assigned to each task, timeline that does not exceed the project due date, resources required. In addition, students **identify** any collaborative tools used. For example, Repl.it, GitHub and so on.



### 2.2. Quality assurance

**Quality criteria**Students **explain** quality criteria based upon the needs from Section 1.1. These quality criteria should contain qualities, characteristics or components that need to be included or visible – based on Section 1.1. – by the end of the current project.

|  |  |
| --- | --- |
| Quality criteria | Explanation |
| Consistent, Accessible and Responsive UI | Every screen must have the same colour palette, typography and spacing. It must be responsive, meaning that it can be viewed on devices with ranging viewport sizes without issue. It also needs to meet W3C accessibility requirements. |
| Robust Scheduling and Booking functionality | There must be a calendar that shows available slots where users can pick a data and time with their details. Though this will rely on constraints on current scheduling API. Though the client should be given a confirmation email, and the practioner will be notified and able to add or update slots and view bookings. |
| Intuitive Navigation | There should be a main menu, that remains at the stop of the screen even when scrolling allowing users to reach any section of the website. |
| Blog Instagram API integration | The blog page must get and display the latest Instagram posts from the practioners website in real time and written longform formatted blog posts made by the Practitioner, with the addition of links to the practioners Instagram page to drive engagement. The web applications own API will need to appropriately handle potential Instagram API issues. |
| About Me Page | The About me page must provide a thorough description about the Practitioner, it must include a photo and credentials, and all these facets should be configurable. The about me page must include contact information such as email and phone number that provide links that opens the corresponding email or phone software. Additionally, a google maps embed should be provided to let users know where the practise is located. |
| Admin Dashboard | The Admin Dashboard must allow the admin (Practitioner) to make changes to the testimonials, about me and blog portions of the website. The Admin changes must update and render to the website for end-users in real-time, with thorough input sanitisation and validation to ensure secure software architecture. |

**Compliance and legislative requirements**Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian [NSW Privacy and Personal Information Act (1998)](https://legislation.nsw.gov.au/view/whole/html/inforce/current/act-1998-133#statusinformation) and/or [Federal Privacy Act (1988)](https://www.legislation.gov.au/Series/C2004A03712). Alternatively, international standards on information security management such as [ISO/IEC 27001](https://www.iso.org/standard/27001) may also be applicable.

|  |  |
| --- | --- |
| Compliance or legislative issue | Methods for mitigation |
| Privacy Act 1988   * Data Handling and Collection | To foster transparency regarding data handling, a clear privacy policy will be implemented into the website detailing what data is collected, why and how it’s stored and used. At the moment, the website will not collect any personal user data. However, as I am using an external service for booking, I’m obligated to inform users as to how they’re data is used. This will include referring them to the external booking service’s Privacy Policy |
| AHPRA   * Advertising a health service | In order to align with the Australian Health Practitioner Regulation agency I must adhere to the following; There will be no misleading or deceptive claims, all will be scientifically backed (hard given the nature of her practises). There will be only verified testimonials, with disclaimers regarding patient consent. The practitioner’s registration number will also be listed on site along with the valid credentials. |
| W3C standards | The website will be perceivable; meaning that the interface will be presented so that users can view the content e.g. captions for images and videos. The website will be able to traversed for a robust user experience, meaning there are clear labels and consistent navigation. For accessibility, the content will be able to be interpreted for those with disabilities, such as compatibility with assistive technology like screen readers. Furthermore, it will adhere to responsive design principles, such as fluid layouts, flexible images and videos allowing for an optimal experience, no matter the viewport size. |

### 2.3. Systems modelling

Students are to **develop** the given tables and diagrams. Students should consult the [Software Engineering Course Specifications](https://library.curriculum.nsw.edu.au/341419dc-8ec2-0289-7225-6db7f2d751ef/94e1eb0a-0df7-4dbe-9b72-5d5e0d17143a/software-engineering-11-12-higher-school-certificate-course-specifications.PDF) guide should they require further detail, exemplars or information. Each subsection below should be completed with Section 1.1. in mind.

**Data dictionaries and data types**  
Students take the needs identified in Section 1.1. of this Systems Report. For each need, students **identify** the variables required, data types, format for display, and so on. (Link to all diagrams

**Admin Login**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 1. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
| Id | Int | NN | 4 | 2 | Unique identifier for administrative users | 1 | Unique positive integer |
| Username | String | XX...XX | 15 | 15 | Administrator’s username | AdminUser | No duplicates  No whitespace No longer than 15 char |
| Password | String | XX...XX | 60 | 60 | Hashed user password | $2a$16$dXaIIWeEt.2iuf6vYAYbYOQlZfTOIfjoTx8uJPxxP1f3qlgTT/oXS | Min 8 Char  1+ uppercase char  1+ lowercase char  1+ number  1+ special char |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Testimonials Page**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 2. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
| Id | Int | NN | 4 | 2 | Unique identifier for testimonial | 1 | Unique positive integer |
| client\_name | String | XX...XX | 50 | 20 | Name of client who consented to provide their positive feedback | Ben J | Non-empty string  No more than 50 chars,  no numbers of special characters |
| Client\_feedback | String | XX...XX | 1000 | 100 | Testimonial body text | Great Service, Better acupuncture | Non-empty string  Max 500 chars |
| Image\_url | String | URL or path | 255 | 60 | Path or URL to an image of the customer for the testimonial | /static/images/testimonials/sarah.png | Must be a valid image path or hosted URL |

**About Me**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 3. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
| Biography | String | XX...XX | 2000 | 200 | Descriptive text about the Practioner and or services | I have been practicing acupuncture for the last 5 years... | Non-empty string  Max 2000 chars |
| Credentials | String | XX...XX | 300 | 300 | The Practitioners credentials | Doctorate in chinese medecine and acupuncture | Non-empty string  Max 300 chars |
| image\_url | string | URL or path | 255 | 60 | Path or URL to image of practitioner | /static/images/pracititoner/HanhLe2025.png | Must be a valid image path or hosted URL |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

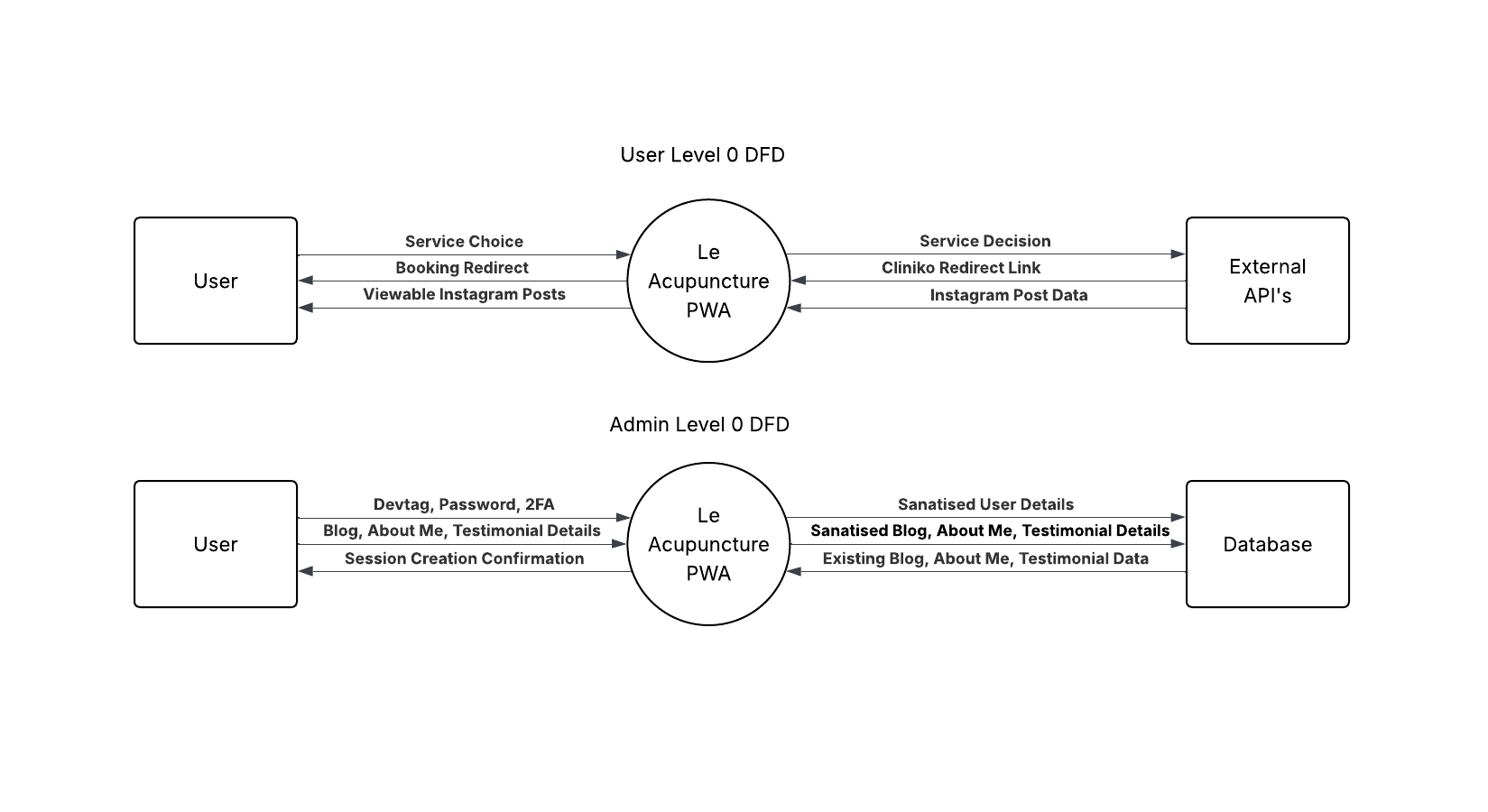
**Blog Database**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 3. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
| Id | Int | NN | **4** | 2 | Unique blog ID | 1 | Unique positive integer |
| title | String | XX...XX | 50 | 50 | URL to banner image | /static/img/banner1.png | Must be a valid image path or URL |
| Blog\_content | String | XX...XXX | 1000 | 800 | The contents of the blog | In the latest news there have been some shocking developments in the acupuncutre world | Non-empty string |
| created\_at | datetime | DD/MM/YYYY HH:MM:SS | 14 | 18 | The time and date the blog was posted | 2025-06-23 13:33:44 | Valid datetime |

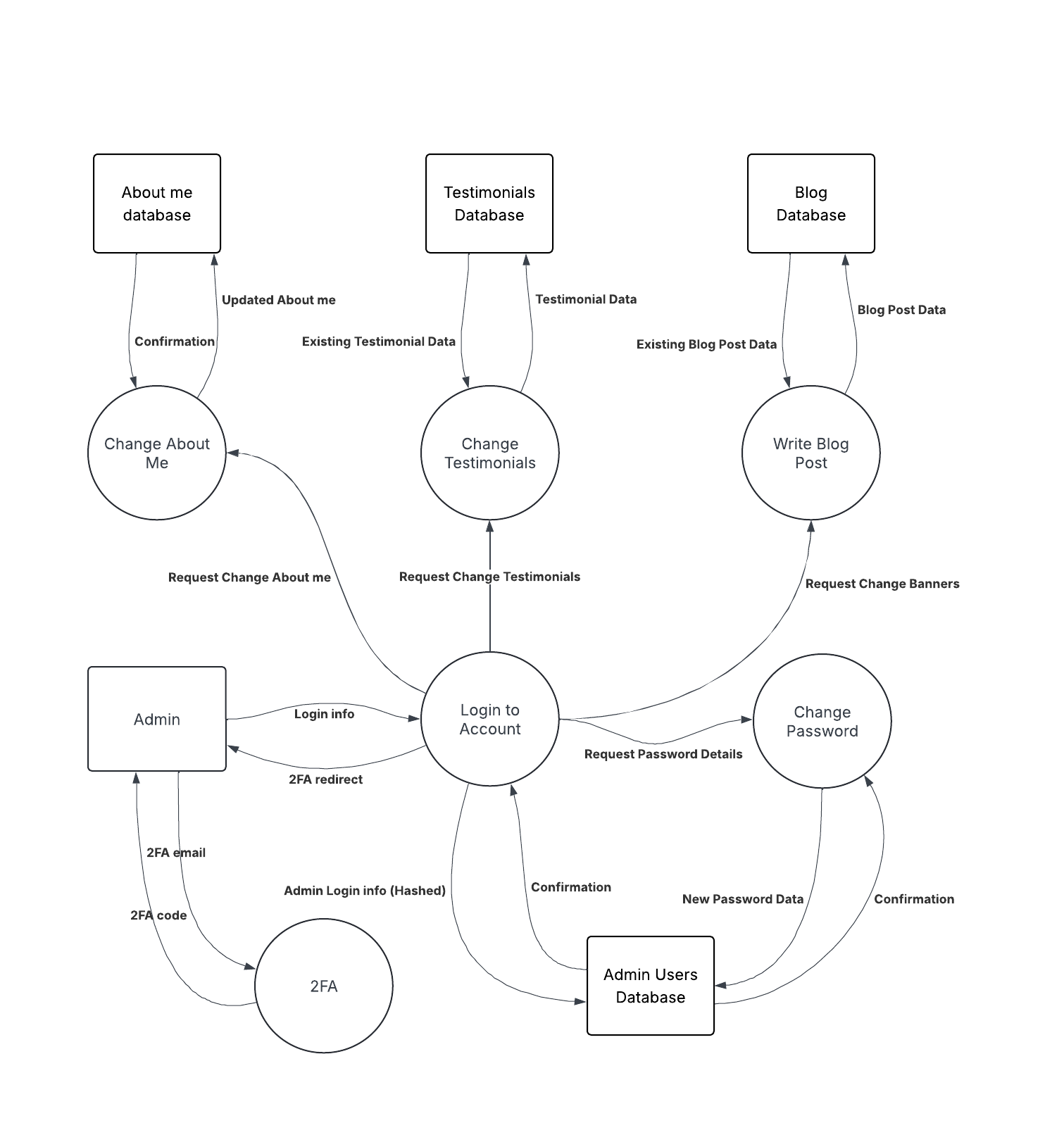
**Data flow diagrams**

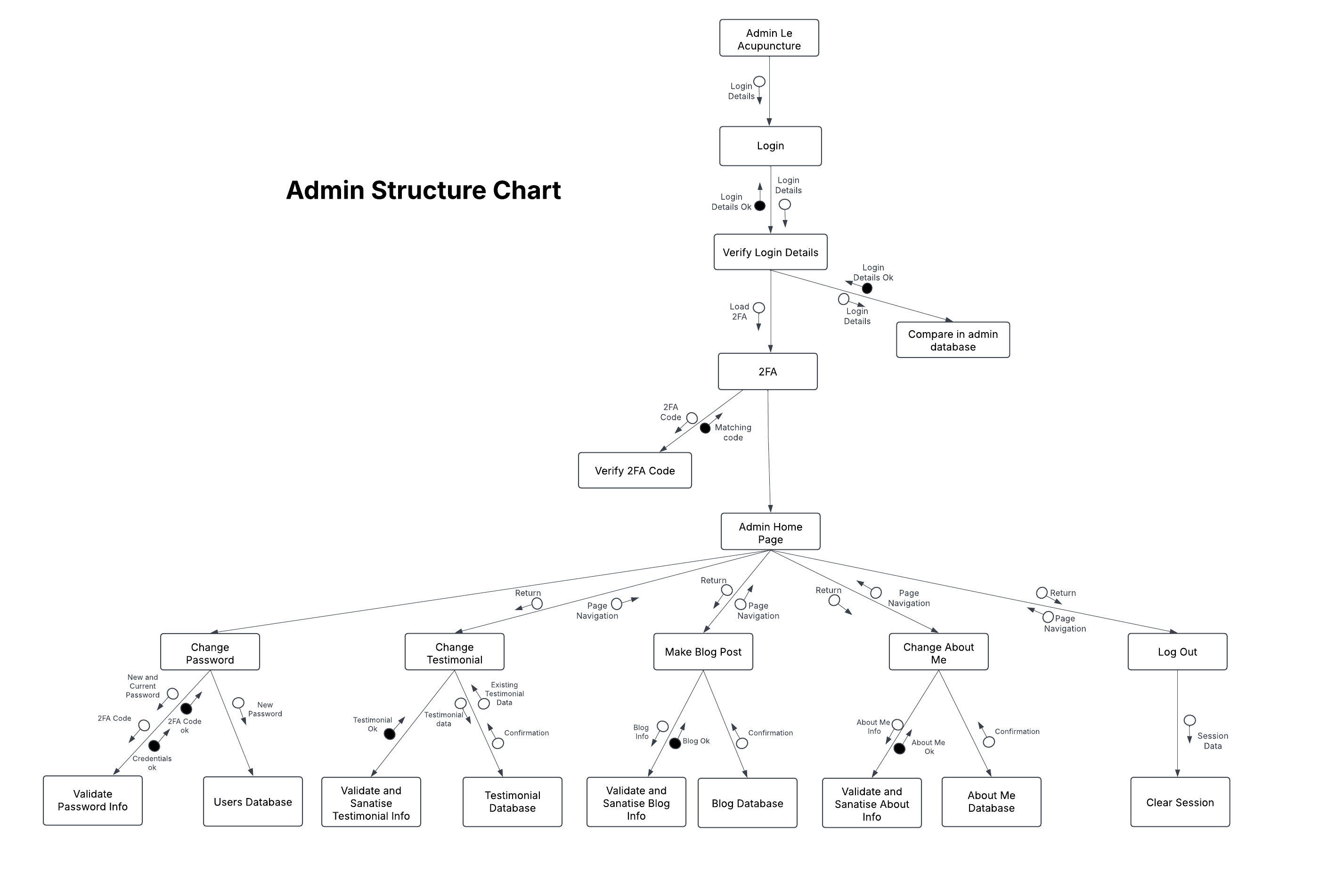
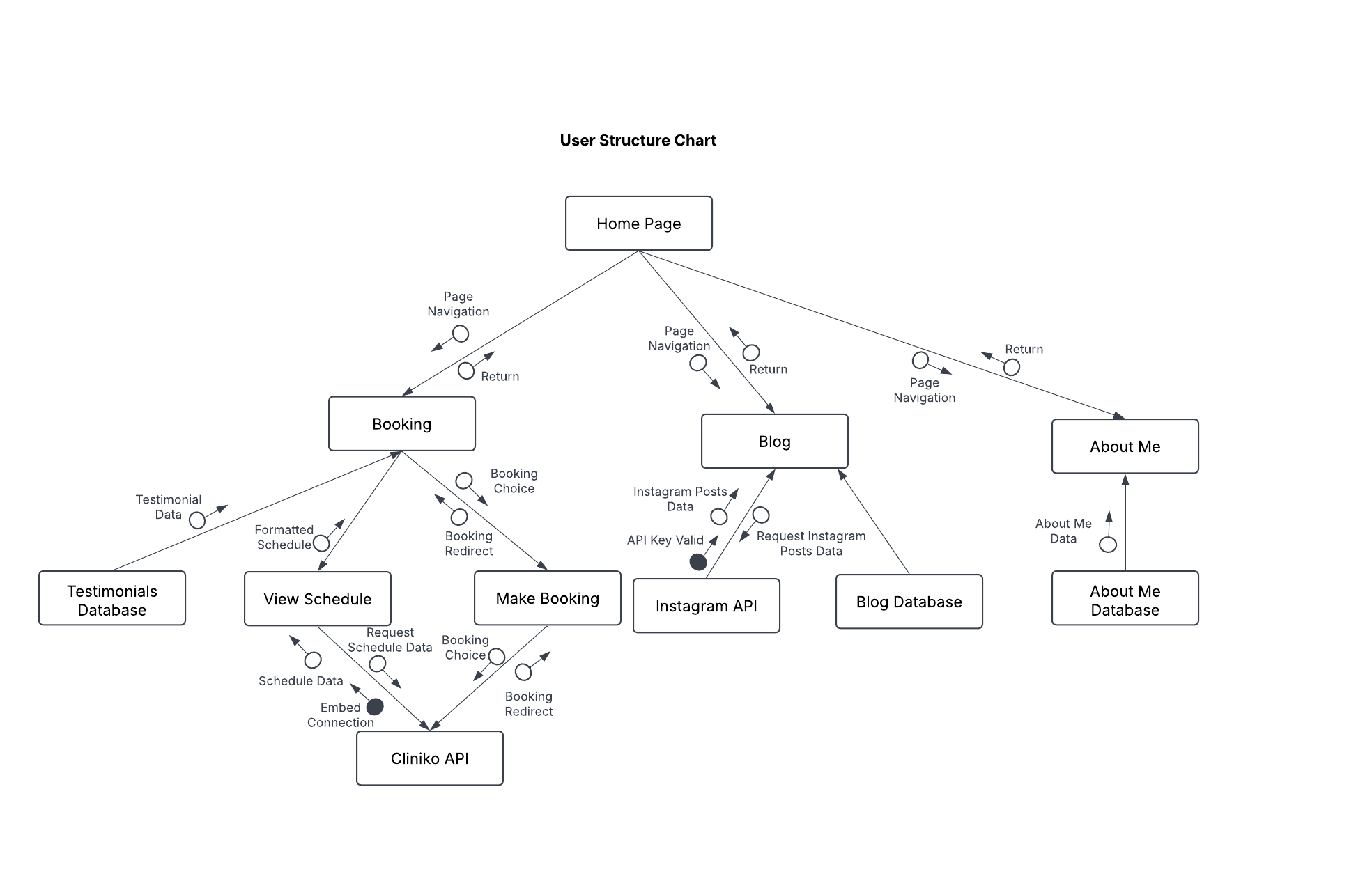
Students **develop** data flow diagrams (DFDs) at Level 0 and Level 1. These diagrams should explicitly include the variables from the data dictionaries previously identified as well as the needs identified in Section 1.1.

*Level 0*

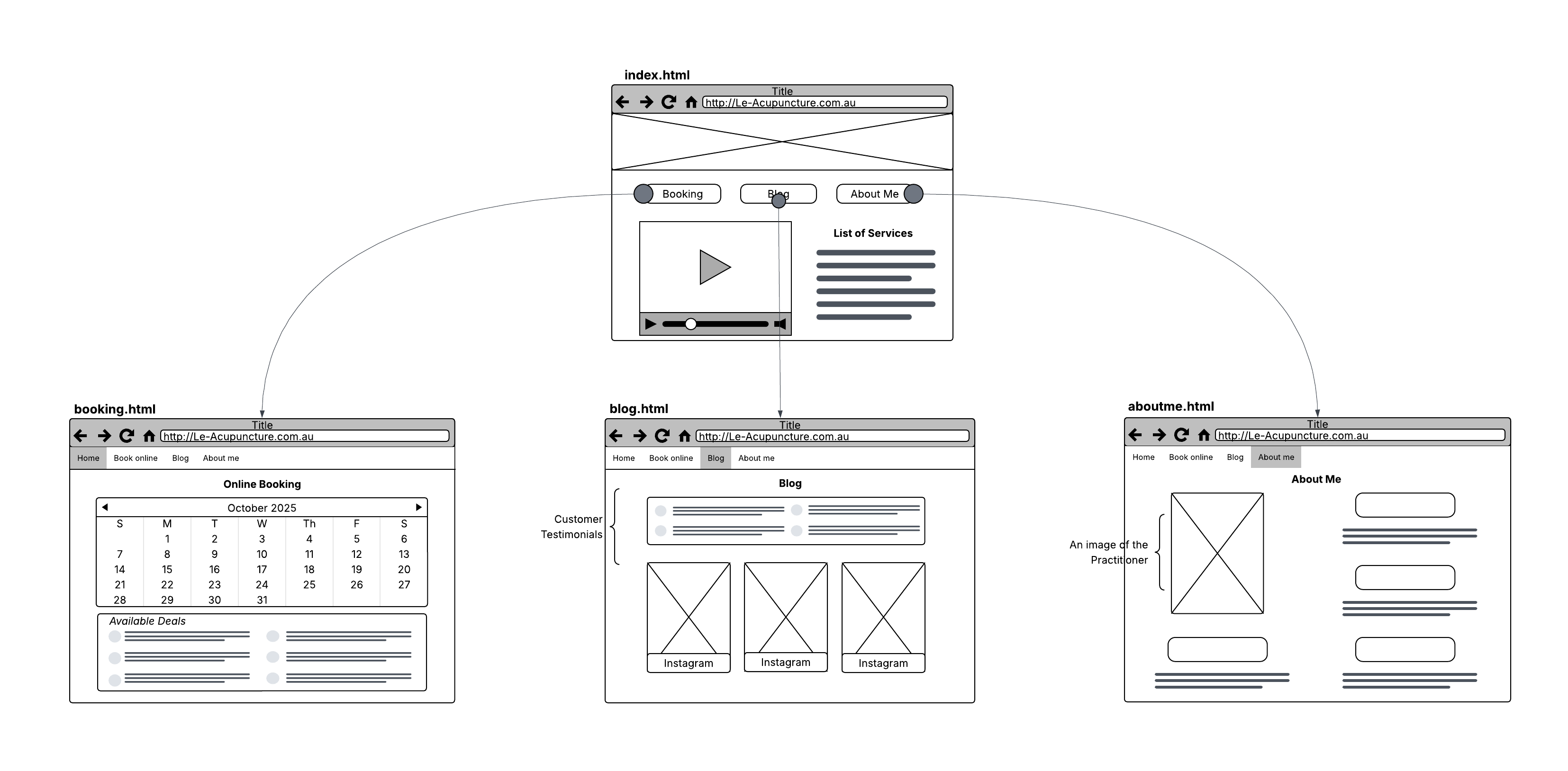


*Level 1 DFD*

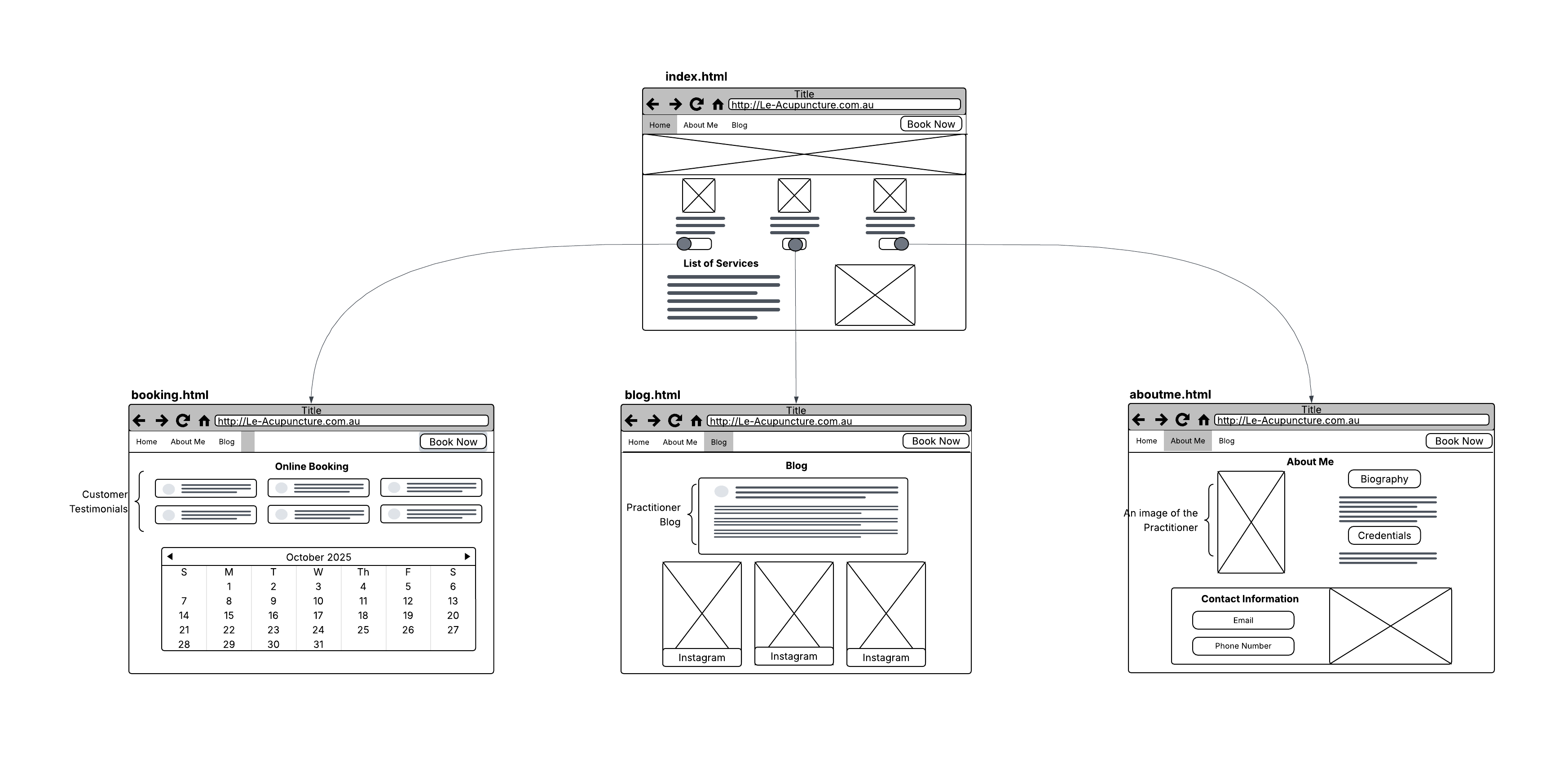


**Structure charts**  
Students **develop** structure charts demonstrating how the procedures, modules or components of the final solutio**Storyboards**  
Students **develop** storyboards, visually representing the software solutions they will build.

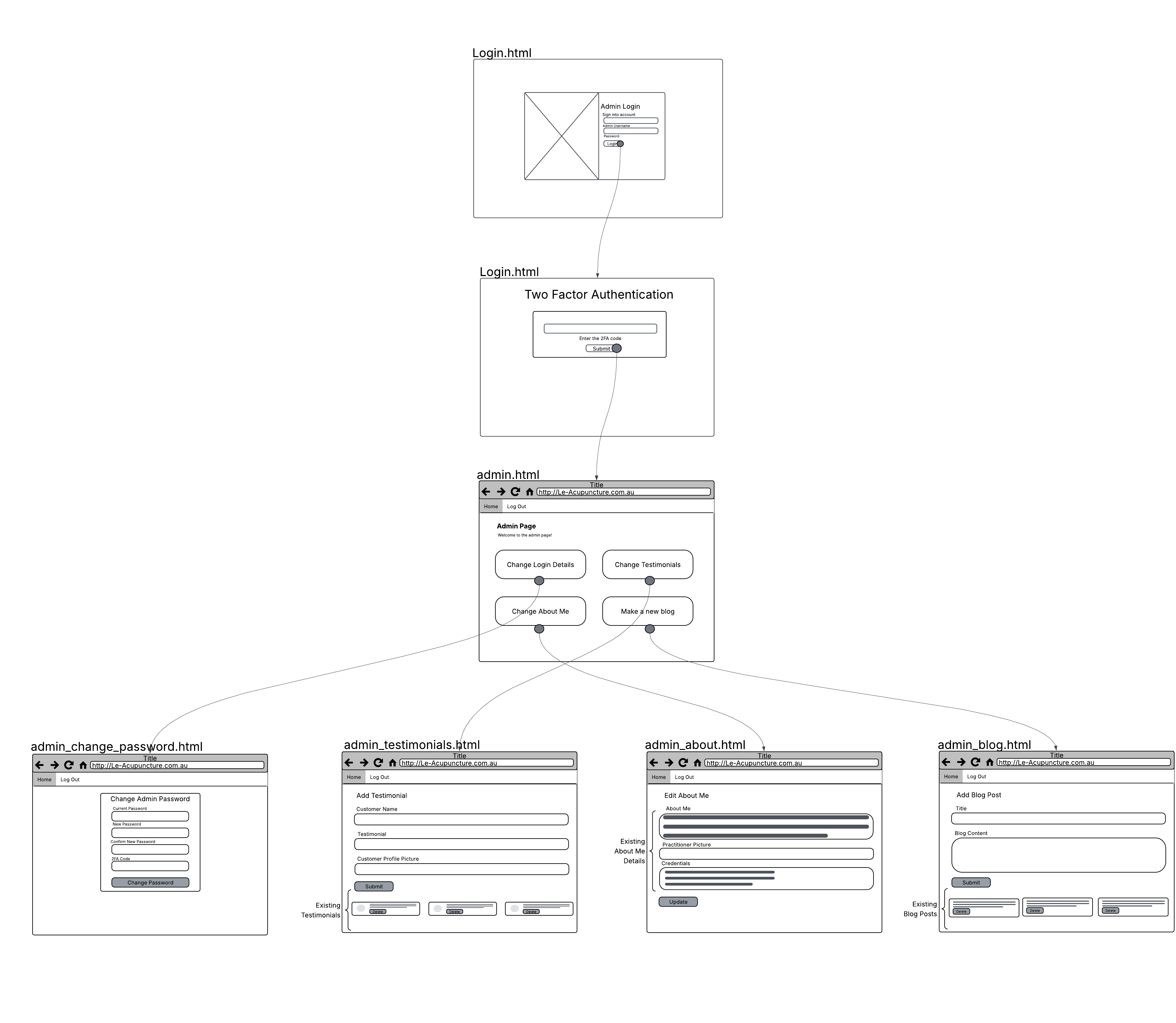
***Initial User storyboard***



**Final User Storyboard**



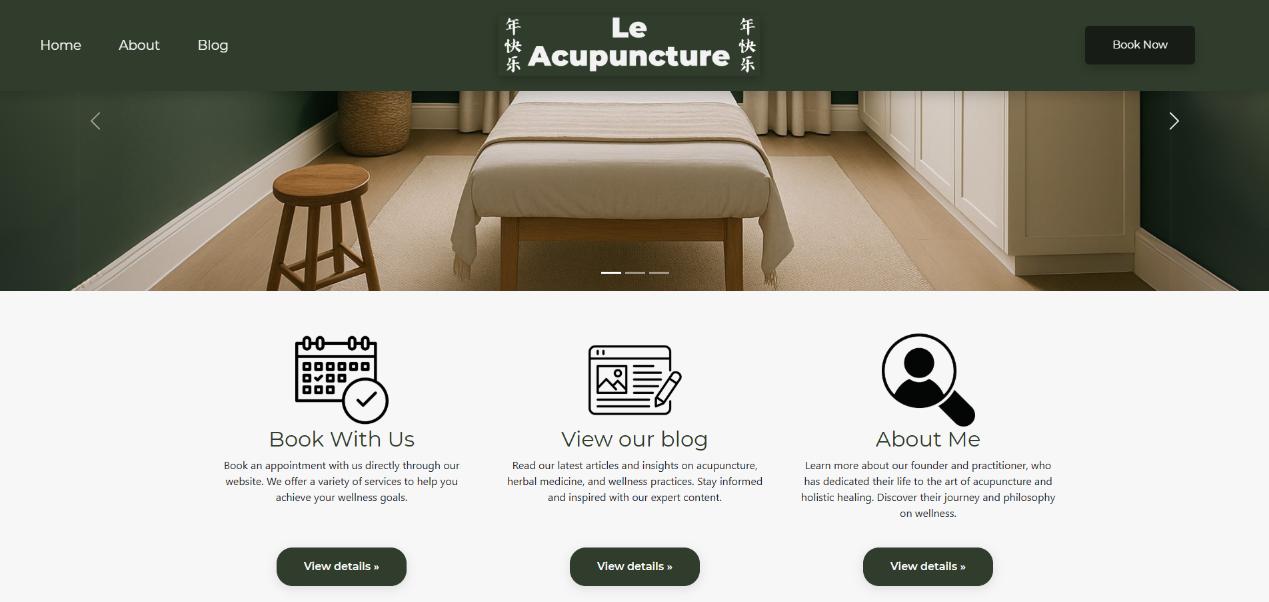
**Admin Storyboard**



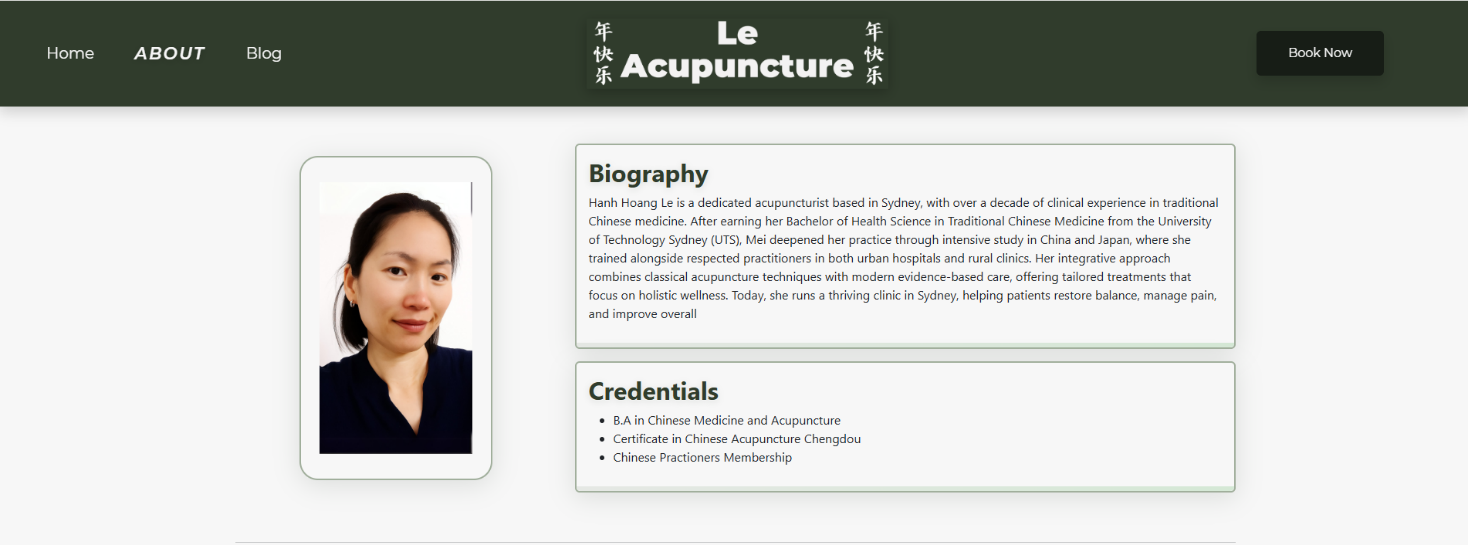
## 3. Producing and implementing

**Solution to software problem**  
Students are to **include** screen shots of their final developed solution here. Each screenshot should include a caption that **explains** how it links to the:

**A clean and effective UI**

* 
* This is the home page of the website, rather the user’s introduction to the website and the practitioner so a clean and effective UI is paramount for the effectiveness of the website. I’ve embodied the practises of clean and effective UI’s through consistency; the website has a consistent dark green and eggshell white colour scheme, which is featured within the carousel images, navbar and the buttons. I’ve also made the navigation intuitive, as from the home page users have multiple ways of getting to the same page and the methods of doing so are clearly illuminated. The UI is also simplistic and unrestricted, meaning that there isn’t visual clutter or too constricting. Cumulatively these allow for an effective UI, as outlined in the client needs.
* **Relation to diagrams + storyboard** 
  + When trying to emulate the home page storyboard design, I realised that the interface appeared constricted, thus I took the ideas of the original such as the buttons to go the different pages and included icons and subtext, to be more accessible and for more visual cohesion.
  + While there are banner images, after discussions with the practitioner, she didn’t feel as it was necessary to customise as she was content with the current images and felt overwhelmed with the additional admin controls.
  + The navbar in the storyboard appeared static, instead I made it responsive and interactive, as when a user hovers over the navbar headings, a line below runs across the length of the heading, and the active page is italicized. Furthermore, I added a book now button on the far right to differentiate it from the other pages, which is common practise in web design to increase customer acquisition.

**About me page**

****

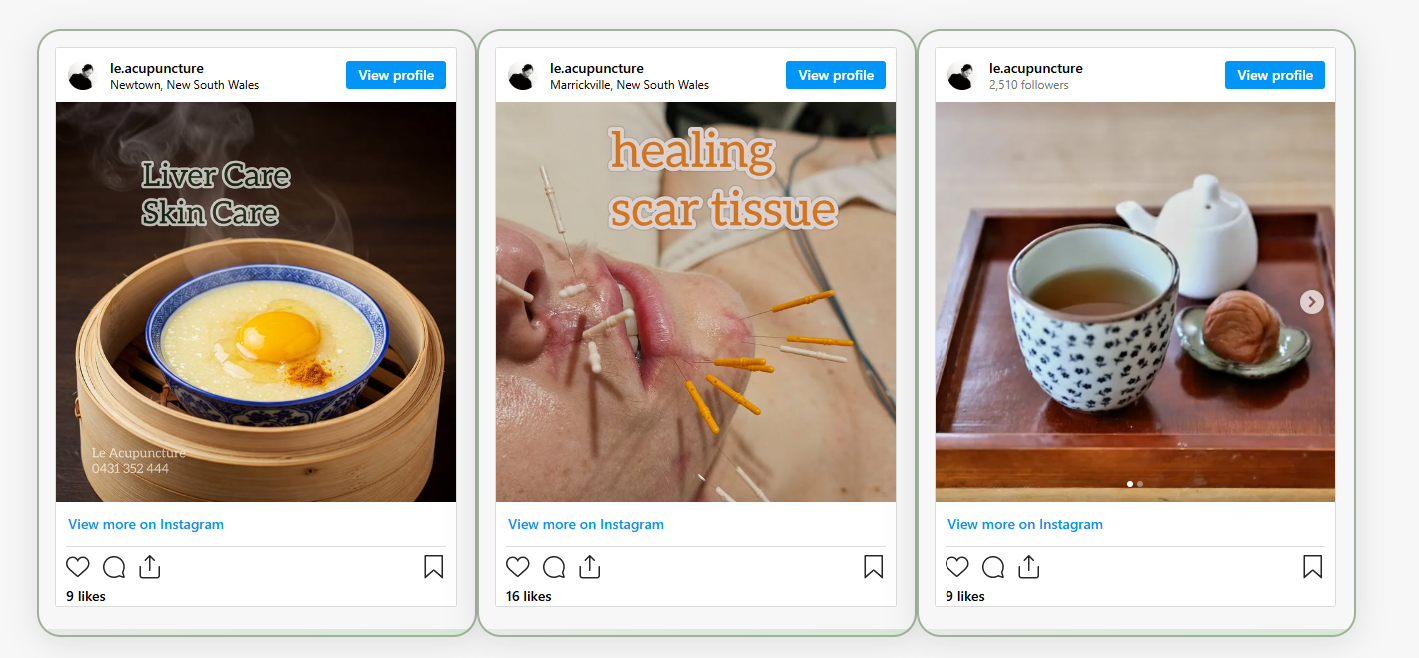
A screenshot of a computer

AI-generated content may be incorrect.

The last website had very limited information about the practitioner. Given the nature of the business, being a one-on-one interaction/exchange, I created an about me page to orientate the users to the practitioner and her credentials. Furthermore, the last website did not provide sufficient information as to where the practise was located or how to get in-contact with the practitioner, so I provided a google embed of the practise as well as bootstrap buttons which when clicked would either open the email or phone app and input the practitioners email or phone number. The about me page allows the Practitioner to change the biography and credentials to match the current circumstances.

* **Relation to diagrams + storyboard**
  + I diverted from the original depictions of the about me page, simply because the storyboard was too constrictive and I wanted the elements to be more spaced out for an effective user experience. Furthermore, the headings and small amount of text felt unnecessary, thus I included a large box for the biography and a smaller one for the credentials next to a photo of the practitioner. Then I included a card below for contact information and google maps embed. The contents of the page align directly with the data dictionary.

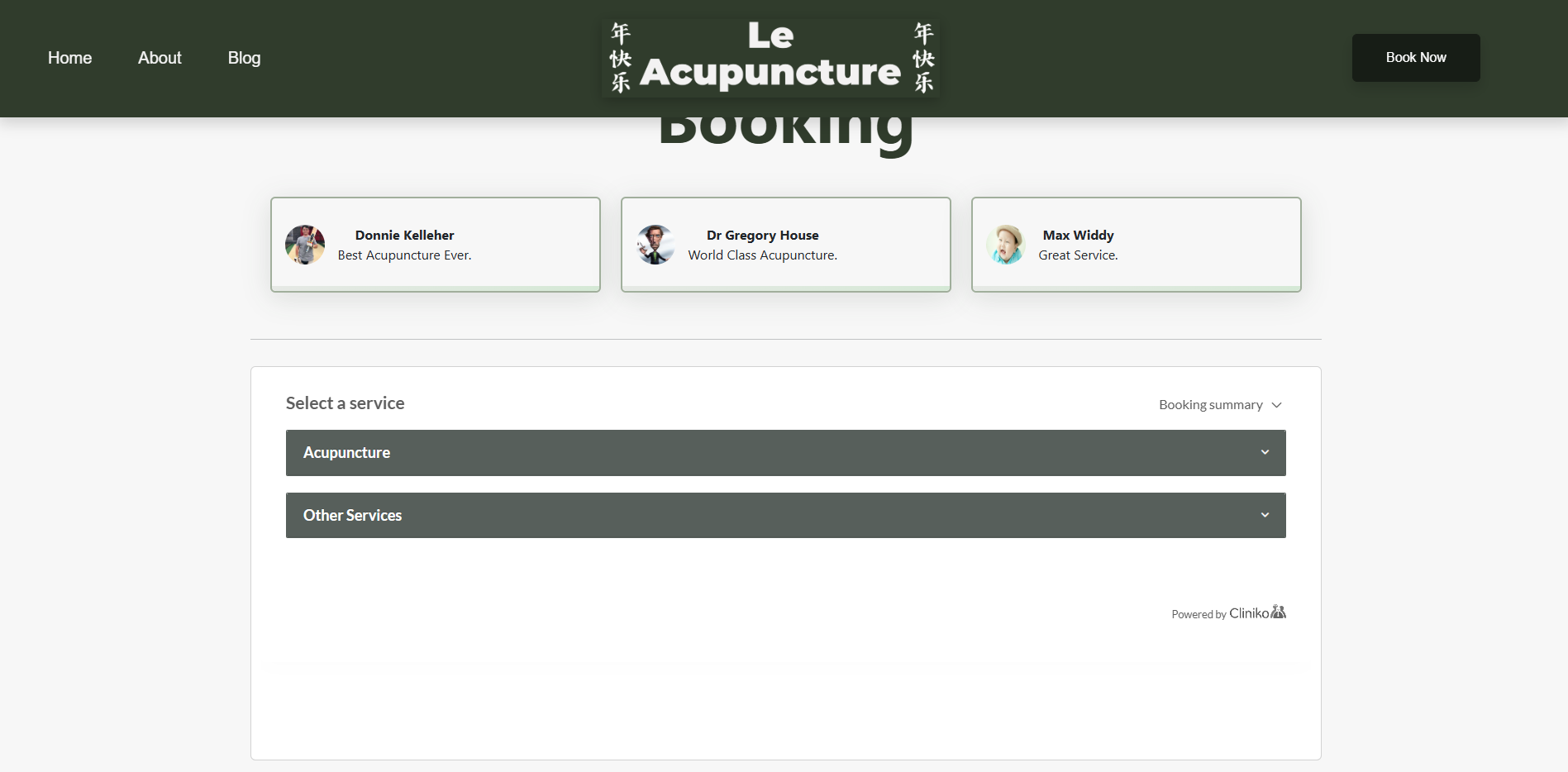




**Blog Page**

The old website did not have a blog page, as discussed previously, I included one to educate viewers on the practise of acupuncture, and provide information on recent developments, with the addition of Instagram posts to drive users to the Instagram page. I also allowed the admin to write the posts with TinyMCE integration for headings and bold text which incorporates the CSS of the website to maintain consistency.

* **Relation to diagrams + storyboard** 
  + The original storyboard had a container which included customer testimonials and below that Instagram API posts. Though upon further review and feedback from the practitioner, including customer testimonials within the blog page did not make sense, the blog page should serve the purpose of being informative not persuasive. Consequently, the customer testimonials were moved to the booking page.
  + I neglected the Practitioners’ desire to construct her own blog posts specifically on recent developments. Thus I included the ability for the practitioner to construct her own blog posts which allowed for headings and special text integration.



**Booking page**

As previously mentioned, the old website did not include a booking feature, I included one on the website which delivers on the following; the user is able to view the Practitioners schedule and book, the Practioner is notified and is able to view the schedule. All of the functionality is handled by Cliniko, a practitioner scheduling service provider which the Practitioner already employs. However, I did add the customer testimonials to the booking page to increase customer acquisition.

* **Relation to diagrams + storyboard**
  + The Cliniko provided the calendar/schedules that is depicted in the storyboard, however after discussing with the Practitioner, she was against offering deals. To mitigate this, I included the customer testimonials on this page instead, right above the booking embed, so that’s the first thing viewers see to increase customer acquisition.

**Version control**Students **describe** what version control system or protocol was implemented.

GIT and agile sprints were implemented for my project as the version control system for my project. Git is open-source version control system which tracks changes to files, allowing developers to phase additions, compare changes, and merge branches. It is a distributed version control system, which differs from centralised version control systems as it did not exist on a single server, allowing me to develop the application on any device with an internet connection. While the Agile Development methodology is a project management approach, I extrapolated the agile sprint procedures as a form of version control. This took the form of creating new sprints, having a product backlog for the website’s functionality, and choosing features to implement within the sprint. Once the sprint was completed, I reflected on the additions made, and my performance in completing sprints to rectify shortcomings and maintain beneficial practices. I then uploaded these changes and merged them to the main branch using the GIT version control system. This allowed to me to create new branches based off the main branch further allowing me to phase in new features.

## 4. Testing and evaluating

### 4.1. Evaluation of code

**Methodology to test and evaluate code**Students **explain** the methodologies used to test and evaluate code. Methodologies include:

* Unit, subsystem and system testing
* Unit Testing
  + Unit testing is the process of testing individual functions or methods within a software application. Each unit (function or method) is isolated and tested and verified it behaves as intended. The main purpose of unit testing is to identify issues before integration into the subsystem which makes up the entire system. The nature of a progressive web application means that the movement spans multiple functions and methods, thus I employed unit testing proactively. For example, in the password verification function, which hashes and salts the user inputted password, I had to verify that the function behaved as intended by hard coding the intended password in the mainline and ensuring that is correctly verified the password.
* Subsystem testing
  + Subsystem testing which can also be described as integration testing, is the process of testing individual pieces of the software application, independent of interactions with other modules. While like unit testing, it rather tests the individual units (functions or methods) that make up the subsystem. The purpose of subsystem testing is to test the individual units within the subsystem interact with each other and behave as intended, as it helps identify issues that arise when components are integrated. This extends to; communication errors, data passing between modules. I utilised sub-system testing when testing the 2FA module, as the movement of data spanned multiple files and functions. I isolated the sub-system and used watches, a debugging tool, to track the movements and transformations of data, such as gathering the 2FA code, verifying the data sent via email etc.
* System testing
  + System is testing is atop the testing hierarchy, where the entire system, consisting of subsystems and within them units, are tested as a complete system. It is the final testing phase before the software is released. The purpose of system is to verify that the software application meets the specified requirements and functions correctly within the intended performance. It comprises assessing functionality, performance, security, and compatibility and made to ensure that it operates as expected within the client specified condition and ready for deployment to end users. System testing can take the form of UAT test scripts which are essentially test cases in which testers are placed in specific scenarios and verify that the software behaves as expected. These test scripts typically span subsystems and encompass all system functionality to ensure effective system testing.
* Black, white and grey box testing
* Black box testing
  + Black box testing is a software testing methodology in which the inner workings of the application is unknown to the tester. It comprises testing functionality without knowledge of logic, against functional requirements, completed during system testing. For my use case, it’s the most effective for client testing as the practitioner does not have any programming skills, thus the tests are executed from their point of view. My integration of black box testing was allowing my client to operate the software as she would normally and assessing if there were any issues.
* White box testing
  + White box testing is another software testing methodology in which the tester has knowledge about the internal structure and code of the software application, and can be employed during unit, subsystem and system testing, though the software application is not executed. The purpose of white box testing is to ensure that the efficiency of code, through testing methods such as control flow and data flow testing. My integration of white box testing was more informal, comprising looking at the internal structure and tracking the movement of data between modules.
* Grey Box Testing
  + Grey Box Testing is the combination of black and white box testing, as it involves testing and evaluating code through both testing the system and evaluating the code. The purpose of grey box testing is to combine the strengths of both box and white box testing which allows for improved test coverage, an optimised testing process, enhanced collaboration between developers and clients, though most significantly it is the most suited for testing web applications as it allows testers to evaluate both the user interface and server-side logic. I utilised grey box testing for security assessments against threats such as invalid forwarding and redirects, which comprised trying to make unauthorised requests and observing the server logs.
* Quality assurance
  + Quality assurance is the series of processes and procedures that ensures a system has been produced within the quality paramters. In the event of shortcomings, there are procedures in place to rectify failures. The initial testing processes include the aforementioned testing methodologies and UAT testing plans. For a progressive web application the most common forms of testing reside in safety, security and accessibility. The predominant procedures that were employed were unit, sub-system and system testing and within them grey box testing was predominantly used. To identify the shortcomings, the version control system and developer documentation were referred to examine what was wrong and where the problems were arising. Quality assurance is evident in my solution, an example of this is when screen readers couldn’t interpret some of the images, so I added alt tags to describe what was happening in the image.

**Code optimisation**Students **explain** the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

* Dead code elimination
  + Removing code (functions, variables, imports and modules) that are never executed, or whose results are never used. It improves efficiency through reducing memory usage and improving maintainability and reusability. I’ve employed dead code elimination through removing unused imports or redundant functions, only including the necessary routes and logic in main.py.
* Code movement
  + Moving code outside of loops or functions if it does not need to be executed multiple times or moving code to a more fitting file. The purpose of code movement is to reduce unnecessary computation. I have employed code movement through abstracting repeated main.py logic into methods.py which is called once per request and then passed to the template rather than repeating the inline logic
* Strength reduction
  + Replacing expensive operations with equivalent but less computationally demanding ones, which can include using in-built functions. My solution does not include any demanding operations, however, to more align with client demands, certain operations such as password validations are efficient built-in operations that were not outsourced to FlaskForm.
* Common sub-expression elimination
  + Identifying and reusing expressions that are computed multiple times, storing their result instead of recalculating, thus increasing efficiecny through reduced computational times. An example of my use case for this optimisation methodology was when assigning the data received from an API endpoint to a variable and passing that through a render template, avoiding redundant function calls and increasing efficiency and readability.
* Compile time evaluation – constant folding and constant propagation
  + Evaluating constant expressions at intial compile time and replacing variables with their corresponding constant values. I utilised this optimisation technique when I defined the auth key constant at the top of the file, so they are set and referred to throughout, allowing Python to optimise its usage.
* Refactoring
  + Restructuring existing code without altering inherent behaviour to improve readability, reduce complexity and improve maintainability. I have applied these methodologies by moving logic and intro seperate files such as methods.py, userManagement.py and testimonials.py. Additonally, my function names and route handlers are concise and clear allowing for enhanced readbility and maintainability.

### 4.2. Evaluation of solution

**Analysis of feedback**Students **analyse** feedback given to them on the new system they have just created. This feedback can be in the form of an interview, survey, focus group, observation or any other applicable method. Students should also include overall positive, negative or neutral sentiments towards the new system in their response.

After conducting an interview with the client the Practitioner, the following were the findings, though keep in mind that the Practitioner lacks technical knowledge thus some of the answers were non-descriptive.

Concerning the User Interface the Practitioner showed a positive sentiment towards the consistent colour scheme, animated buttons and the ease of navigation. Though the Practitioner reacted negatively to the bold logo and pointed out that the Chinese characters said happy new year, and Chinese customers would be confused.

Concerning the home page the Practioner responded positively to the carousel and the integrated buttons though thought that the featurettes looked bleak and the list of services should’ve been more descriptive and provided links to accreddited descriptions.

After reviewing the About Me page, the practitioner showed a positive sentiment towards the visual clarity, and the email and phone number integration that opened the corresponding email and phone number applications. Additionally, the Practitioner liked that the about me content could be changed in real-time. However, she argued that the headlines for the About Me, Biography and Credentials were unecessary, this is was a repeated sentiment.

Regarding the blog, the Practitioner appreciated the Instagram post integration, and that she could write blog posts with proper formatting and delete existing blog posts. Though, as mentioned previously, she thought that the blog heading was unnecessary and repetitive.

For the booking page, the Practitioner responded positively to the configurable testimonials and the Cliniko integration which she already utilises. However, she did not respond positively to the absence of price listing for different services, which can be portrayed as more of a reflection of the Cliniko integration,

Concerning the Admin dashboard, the Practitioner showed purely positive sentiments, pertaining to the simplicity and intuitiveness, and the fact it allowed real-time changes across the majority of the pages of the website.

**Testing methods**Students **identify** the method or methods of testing used in this current project. For each they use, students are to **explain** how and why it was used.

|  |  |  |
| --- | --- | --- |
| Method | Applicability | Reasoning |
| Functional testing | Functional testing is the process of testing individual functions or features that they work as specified through input and output verification. I implemented this in all my API calls through the methods functions which use exception blocks to test HTTP responses. | I utilised functional testing to mainly  ensure that API integrations and  endpoints were working reliably. The  usage of the application relies heavily  on API calls, thus every function was  tested to ensure it handles both  success and failures properly, to  prevent crashes and ensure the  intended behaviour. |
| User Acceptance testing | User acceptance testing is the final stage of software testing where real users or a developer testing teams validate if a system meets their needs and is ready for realease. I applied user acceptance testing by creating a UAT testing document which can be viewed in the attached documentation, which outlines objectives, scope, testing team, requirements and test scripts. | I utilised user acceptance testing in order to confirm that the solution aligned with the clients' expectations with the test scripts testing the core client needs. This included testing end user scenario such as booking and navigating the website. Aswell as admin controls for creating blogs and altering the about me. |
| Live data | Live data testing is the process of testing the system with actual production data from real databases and systems, using genuine user-generated content rather than artificial test data. Though currently I have not employed this testing methodology as the practitioner has not provided sufficient data as of yet due to a busy schedule. Thus, the data retrieved from the API and databases are largely gathered from external sources, meant to simulate the live data that would be present on the final implementation of the website. | As mentioned previously I have not provided live data due to the Practitioners busy scheduler, rather simulated data that is used to test the system before the authentic data is provided on final release. Though authentic data was used in the about me portion of the website. |
| Simulated data | Simulated data testing is when the  system is tested with artificially  constructed data, gathered from  external sources or via computational  methods. I employed this method  when creating simulated testimonials  and blog data. | I used simulated data in order to  validate and test the complex  innerworkings of the solution before  applying them to authentic data. I  needed to use simulated data, as to  use authentic data would lower  efficiency due to practitioner oversight. |
| Beta testing | Beta testing is a process of allowing  real-users to test the software  application before release. I allowed  the client (Practitioner) to operate the  Website in a manner they would  normally use it in order to uncover  bugs or vulnerabilities. | Beta testing uncovers usability issues,  and performance issues under real  usage patterns. An example of its benefit is when another end user tried to book through the website and the embed took far to long to load, these I moved the JavaScript to the app.js thus rectifying the performance issue. |
| Volume testing | Volume testing is the process of testing  system performance and stability under  high load. I applied this testing procedure through manual continuous API calls, authorisation keys and API limiters to verify that the system can handle expected and peak system traffic without degrading performance or crashing. | I employed volume testing to verify that the system had protection against system overloads. My findings were that the authorization method and api limiters provided a defence against unauthorized API abusage. Important to note that more comprehensive volume testing will need to be utilised when there are a high number of concurrent users. |

**Security Assessment**Students are to **perform** an extensive security assessment of their final application and e**xplain** the countermeasures implemented.

|  |  |
| --- | --- |
| Threat | Countermeasure |
| Cross Site Scripting (XSS) | A Cross site scripting (XSS) attack, involves a threat actor injecting malicious code into an unsanitised input within a website which is stored on a server, allowing attackers to compromise a user’s interactions with the site.  For e.g. A user makes the admin places the following script in a blog <script>www.malicious.com</script> and if unsanitised it executes when a user loads the page (possibly allowing a hijacked session)  I implemented input sanitization and a content security policy to block potential cross-site scripting attacks. All users' inputs are validated and sanitized using the Bleach library to strip more complex code addition inputs of harmful HTML characters, additionally when blogs are created the tinyMCE editor passes in HTML characters which are removed manually during the sanitization process.  While the content security policy strictly defines the content that is allowed to be executed on the website. Though I do include <iframe> and <scripts> the sources are defined in the CSP preventing XSS attacks.  Input sanitization minimizes the risk of malicious code injection, as the malicious characters are removed, preventing them from being identified as executable code within the solution. If it were to be bypassed somehow, the inline scripts would be blocked by the Content Security Policy. |
| Cross Site Forgery Request (CSRF) | A cross-site forgery request (CSRF) attack is when a threat actor forces an end user to execute unwanted actions within an application that they’re already authenticated with, usually through a link or email. They then perform state-changing requests such as changing emails and passwords.  *For e.g. A user, logged in with bank.com, receives a link for a website with this code*  *“<img src="https://bank.com/transfer?amount=1000&to=attacker" />”*  Flask’s CSRF protect function requires a CSRF token which is a random, unique string, assigned to a user once they login. They are validated for all state-changing requests, as the user inputs it as a hidden field in every form. Thus, the CSRF token counters any CSRF attacks as if the admin clicked a link that would’ve made a state-changing request, it would be denied as the once the webpage loaded the CSRF token would be missing. |
| Invalid Forwarding and Redirecting | Invalid forwarding and redirecting is where a user accepts an untrusted input, such as a trusted domain name with a modified link, causing the web application to redirect to the threat actors unsafe site, mirroring a phishing attempt. For e.g when user clicks on le-acupuncture.com/admin?url=http://malicious.com they’re redirected to malicious spoofed website  The countermeasures I’ve implemented against invalid forwarding and redirecting is the Content Security Policy and a list of whitelisted URL’s. The CSP outlines the allowed URL sources, only allowing connections coming from the origin, preventing redirects to external API or websites. Whereas the whitelisted URL’s function outlines the only allowed public routes. Cumulatively, these countermeasures prevent invalid forwarding and redirecting as the CSP and the Whitelisted URLs define the allowed destinations, invalid forwards and redirects and automatically redirected back to the home page. |
| Race Conditions | Race condition vulnerabilities occur when a program has multiple threads accessing and executing shared data simultaneously. When mismanaged, it can lead to unpredictable outcomes creating a race window where threat actors can gain exploit a system either by corrupting data, gaining authorized access or causing a system crash. The countermeasures I’ve implemented are thread synchronization and rate limiting. Thread synchronization is achieved through the CSRF protect function which renders operations between threads consistent, synchronizing threads to ensure only one request is handled at a time. While rate limiting, which is defined within every API endpoint, limits the number of requests that can be made. The thread synchronization eliminates timing difference, preventing a race condition exploit by ensuring username and password checks are inseparable. Rate Limiting mitigates heavy loads that could exhaust the server resources that could lead to loss of data or exploitation. |
| Broken Authentication and Session Management | Broken authentication and session management occurs when threat actors exploit weaknesses in how users are identified, verified and tracked by the application. Vulnerabilities include, allowing weak credentials and session ID’s, insecure caching and sessions that do not expire. To mitigate this security threat, strong password’s were enforced and secure session management was implemented. A manual password validation function was implemented which included; lower and uppercase letters, numbers and special characters. Additionally, Flask sessions were implemented, stored using secure cookies, which were configured to have a 24 hour session lifetime and did not cache sensitive data. Password enforcement and secure session management, effectively mitigates both dictionary attacks and the theft of session data, acting as a robust sheild against broken authentication and session management. |

**Test data tables**Students **identify** variables which were used for either path and/or boundary testing. Students **develop** these test data tables based on their algorithms versus their real code. Students then **state** the reason for including said variables.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variable | Maximum | Minimum | Default Value/Example | Expected Output | Actual Output | Reason for Inclusion |
| New\_password (input) | 128 | 8 | P@ssword123 | {'length': False} if ≥8 | {'length': True} if <8 | Boundary tests minimum password length |
| Title (blog input) | 200 | 1 | Latest Developments 2025 | Santitized title | Not null input | Boundary testing for length overflow |
| blog\_content (blog input) | 10000 | 1 | In the latest Acupuncture developments... | Sanitizied blog content | Not null input | Boundary testing for blog content limits |
| Biography (about me input) | 5000 | 10 | Hanh Le has a Doctorate in chinese medicine... | Sanitized bipgraphy | Not null input | Boundary testing for biography limits |
| Customer\_testimonial (testimonial input) | 2000 | 1 | Great  Acupuncture,  better people | Sanitized  testimonial | Not null input | Boundatry tests testimonial content limits |
| filename (about me + testimonial input) | 255 | 5 | Tony\_Soprano.png | File validation result | 0 or 1 from  allowed\_file() | Boundary testing to prevent system path errors and filename limits |

**Analysis of solution against quality success criteria**Students are to take each quality success criteria from Section 2.2 and place it here. For each quality criteria, **analyse** the components of the solution that met or did not meet each quality criteria. Give reasons why each success criteria were or were not met.

|  |  |  |
| --- | --- | --- |
| Quality criteria | Met? | Analysis |
| Consistent, Accessible and Responsive UI | Yes | The User interface has a consistent appearance and font pairings, a robust colour scheme and allows for internationalisation given the user has the google translate widget. The User interface is responsive and can be viewed on different devices with differing viewport sizes and follows the W3C standards of security, privacy, web accessibility and internationalization |
| Robust Booking Page | Yes | The booking functionality is provided by the Cliniko service which the practitioner already uses. Thus, it has met the criteria as users can access a calendar that displays available time slots, users can book and the Practitioner is notified and can view the schedule. |
| Inuitive Navigation | Yes | The Navbar is responsive and allows users to navigate to any portion of the website. |
| Blog + Instagram API integration | No | While there are Instagram posts from the Practitioner on the website, they are embeds and there is no Instagram API integration. This is because the Meta Developer Suite requires websites to be published and approved before allowing Threads API integration, additionally the Meta Developer Suite docs surrounding integrating Instagram posts is convoluted and I was unable to use the Instagram (Threads) API to integrate it into the website. However, the Practitioner can create formatted posts which can be uploaded to the website in realtime. |
| Descriptive About Me Page | Yes | The about me page provides an indepth biography about the practitioner, her credentials and contact information, such as email and phone number and when interacted with open the corresponding apps. There is also a google embed feature to show users where the Practitioner’s clinic is situated. |
| Admin Dashboard | Yes | The Admin Dashboard allows the Practitioner to make real-time changes to the testimonials, blogs and about me page with secure validation and sanitisation. |