
**BIOM9450 Biomedical Informatics**

*Major Project Report*

# **NutriMed:** A Web-based Medication and Diet Regime Management System

***Group: PALM***


| | |
|---|---|
| *Aayushma Lohani* | *z5310052* |
| *Michelle Kaminski* | *z5362642* |
| *Paris Briggs* | *z5312301* |
| *Laila Abdul Waheed* | *z5527270* |


**T3 2024**

**THE UNIVERSITY OF NEW SOUTH WALES**
**GRADUATE SCHOOL OF BIOMEDICAL ENGINEERING**

# Group Plagiarism Declaration

| BIOM9450 | Biomedical Informatics | 30/11/2024 |
|---|---|---|
| *Course ID* | *Course Name* | *Date* |

| T3 2024 | Major Project Final Report |
|---|---|
| *Term & Year* | *Title of Assessment* |

| Hamid Rokny |
|---|
| *Lecturer* |

**We the undersigned, individually declare that:**

- the percentages listed below represent a fair indication of the proportional contribution made by each member of the group submitting this piece of work for assessment;
- my contribution to this assessment item is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere;
- all reasonable care has been taken to ensure that no other person has been able to copy this work either in paper or electronic form.

**We acknowledge that the assessor of this item may, for the purpose of assessing this item:**

- reproduce this assessment item and provide a copy to another member of the University; and/or,
- communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).

**We certify that we have each read and understood the University rules in respect of Student Academic Misconduct.**

| Student Number | Student Name | Percentage |
|---|---|---|
| z5310052 | Aayushma Lohani | 25% |
| z5362642 | Michelle Kaminski | 25% |
| z5312301 | Paris Briggs | 25% |
| z5527270 | Laila Abdul Waheed | 25% |

This report details the tasks undertaken, approaches adopted, and the functionalities implemented for the major project component of the BIOM9450 course. It highlights the design and development processes, including database creation, user interface design, dynamic content integration, security implementation and user validation features. It also includes a user manual for testing the project (accessed in the zip file).

## Understanding the Project

The project began with a thorough review of the assignment specifications to fully understand the project goals, deliverables, and expectations. This step was crucial in aligning team objectives and individual efforts.

We began a series of weekly meetings to discuss our thoughts and progresses. In the first meeting, we outlined the scope of the project, determined the starting point, and made key decisions regarding timelines and task management. We broke the project into smaller, achievable tasks and assigned initial responsibilities based on individual strengths and performances. During our meetings, we not only discussed our current progress, but also our goals and tasks for the upcoming week, so that we remained organised and on-track with our assigned timelines.

As a starting point, we discussed what our general webpage would look like and the features it would contain (e.g. pages to include, programming language used, etc.) so that begin our forming initial webpages to start the project. Individual task allocations were assigned during the Week 1 Tutorial, although we collaborated and supported each other to complete the project.

**Table 1.** Task Allocation

| *Aayushma* | *Michelle* | *Paris* | *Laila* |
|---|---|---|---|
| Normalisation, Website user manual | SQL error handling | Code | Report, Justification |
| Data type selection, PHP server validation | Database (20) | Web functionality | User interface |

## Version Control

To facilitate collaboration and prevent conflicts in code, we ensured that Git was properly configured on all our laptops. A shared Git repository was created, with each team member working on a separate branch. This approach allowed for an isolated workspace for individual tasks, enabling efficient version control and reduction in conflicts during the development stage. Commit conventions and branching protocols were agreed upon to ensure clarity and traceability.

## Key Features of our System

The NutriMed software supports several practical features designed to streamline healthcare management efficiency in an aged-care facility. These features ensure efficiency, security, user-friendliness and personalised care for both practitioners and administrators.

The platform provides a personalised experience for logged-in users through dynamic session management. Upon login, all users are greeted with a "Welcome [username]" message displayed in the right-hand corner of all the pages, creating an engaging and user-centric interface.

One of the standout features of the platform is the ability to generate PDF reports for both practitioner and patients. These reports consolidate vital information about medical histories, medication rounds, and diet planes, in a professional format. This function is designed for record-keeping and sharing details insights with stakeholders.

To ensure accountability and proper oversight, the platform automatically sends an email to the facility director when a patient refuses a medication or diet round. This immediate notification systems helps address potential issues promptly, ensuring that patient care remains a priority.

Practitioners cannot self-register through the login page; instead, new practitioners must be added by an administrator. This approach ensures that only verified personnel gain access to the system, minimising any security risks.

All the features have been clearly outlined in the provided user manual.

## Development Process

*Phase 1: Database Design and Development*

The first step was to determine what data was essential and needed to be formed to support the website's core functionalities. By examining the specifications for the project, we decided that patient records, practitioner details, medication schedules, dietary regimes, and user login information would be required. Each data type was reviewed meticulously, to ensure that no critical information was overlooked. For instance, medication schedules and dietary plans needed to align with patient-specific conditions.

Once these data fields were identified, we grouped related information into categories to avoid duplication and ensured database normalisation. For instance, login credentials and user details were stored securely in a user's table to manage access control. These login credentials (in specific the passwords) were later hashed to ensure webpage security. This approach reduced redundancy, improved data consistency, and ensured the database would be scalable and maintainable.

With the data fields grouped, we designed the database schema using MAMP (a local development environment), detailing each table's structure, primary keys, foreign keys, and data types. To give an example, the Medications table captured MedicationID (primary key), PatientID (foreign key), and prescribed medications.

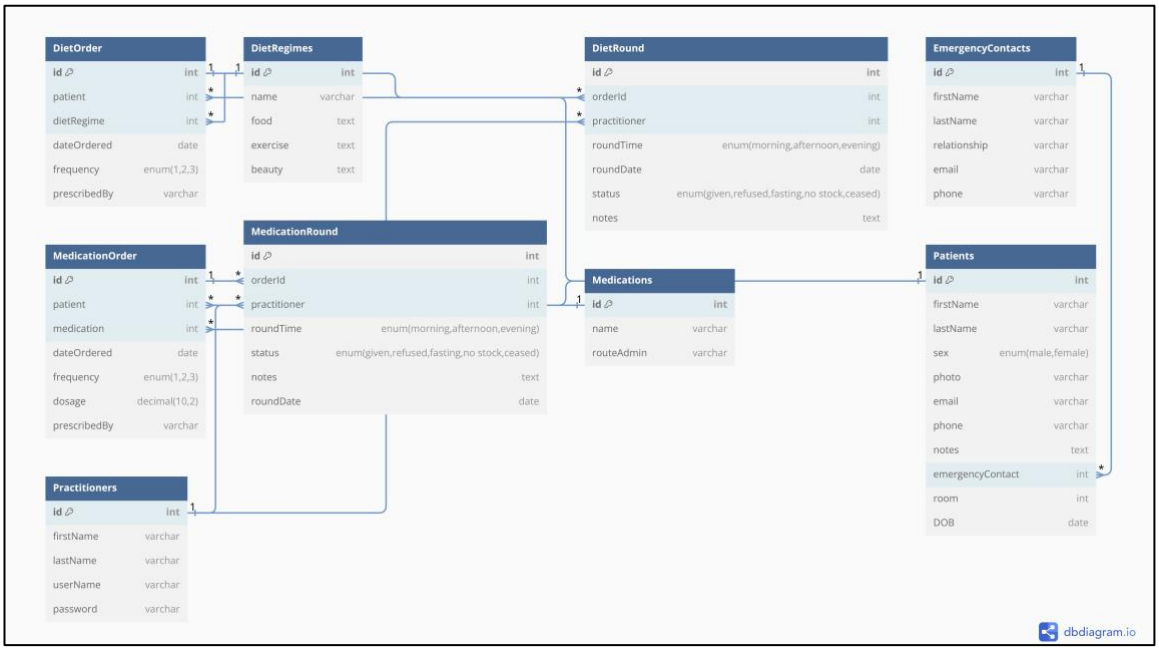


***Figure 1.*** *Database relationship diagram.*

An example of the DietRound table that manages patient diet rounds effectively, including its constraints and relationships:

```
CREATE TABLE `DietRound` (
  `id` int NOT NULL,
  `orderId` int DEFAULT NULL,
  `practitioner` int DEFAULT NULL,
  `roundTime` enum('morning','afternoon','evening') DEFAULT NULL,
  `roundDate` date NOT NULL,
  `status` enum('given','refused','fasting','no stock','ceased') DEFAULT NULL,
  `notes` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

ALTER TABLE `DietRound`
  ADD PRIMARY KEY (`id`),
  ADD KEY `orderId` (`orderId`),
  ADD KEY `practitioner` (`practitioner`);

ALTER TABLE `DietRound`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=379;

ALTER TABLE `DietRound`
  ADD CONSTRAINT `dietround_ibfk_1` FOREIGN KEY (`orderId`) REFERENCES
`DietOrder` (`id`),
  ADD CONSTRAINT `dietround_ibfk_2` FOREIGN KEY (`practitioner`)
REFERENCES `Practitioners` (`id`);
```

*Phase 2: Populating the Database*

To ensure the database was functional, we populated it with realistic sample data, inspired by common medical conditions and dietary encountered in aged-care facilities (elderly population). Research was conducted into the general dietary requirements for elderly individuals, as well as common disorders they may experience. For example, a resident with Type 2 diabetes, would require insulin injections before each meal and benefit from a diet consisting of low glycaemic-index foods. Aligning diet with prescribed medications ensure that any side effects are not obtained, and the general health of the patient is improved. Correlating diet with medications are required in real-life situations (to prevent contraindications), and this inclusion in our sample data enabled us to test the database's performance under realistic scenarios.

```
INSERT INTO `DietRound` (`id`, `orderId`, `practitioner`, `roundTime`, `roundDate`, `status`, `notes`) VALUES
(1, 1, 1, 'morning', '2024-11-28', 'given', 'Fresh fruit and unsalted oatmeal served. Patient ate well.'),
(2, 2, 1, 'afternoon', '2024-11-28', 'refused', 'Vegetable soup refused. Patient cited dislike for spinach.'),
(3, 3, 2, 'evening', '2024-11-28', 'fasting', 'Patient on fasting for blood test.'),
(4, 4, 2, 'morning', '2024-11-29', 'given', 'Oatmeal with bananas consumed.'),
(5, 5, 3, 'afternoon', '2024-11-29', 'no stock', 'Requested low-sodium crackers unavailable.');
```

The above code employs INSERT INTO to add initial rows into the DietRound table, and shows realistic entries, such as dietary notes.

*Phase 3: Dynamic Queries and Updates*

As the website's functionality evolved, the database schema required modifications. For instance, when new features, such as the generation of practitioner-specific reports, were added, new fields were introduced to store the necessary data. Queries were written to support the specific needs of each webpage. For example, medications and diets regimes needed to be filtered based on a specific patient when forming patient reports. Each query was tested to ensure accurate data retrieval and integration with the frontend. For each functionality (e.g. generating patient reports), we tested it to make sure that patient reports could be generated for each patient and similarly, for each practitioner for the practitioner reports.

One of the key challenges during the database development was ensuring that any changes to functionality did not disrupt the existing schema. To address this, we maintained an iterative approach and regularly documented the updates to the schema and queries. We comprehensively tested the database after each change to identify and resolve any issues promptly. Additionally, integrating the database with the PHP backend presented challenges in maintaining consistency across pages. These were resolved through careful debugging and validating each database operation.

The final database was a robust and scalable solution capable of handling the project's requirements. It supported secure data storage, efficient retrieval, and dynamic integration with the website, laying the foundation for a functional and user-friendly application.

**Website Design and Development**

The development of the website followed a very structured approach, starting with beginning with initial mock-ups, progressing to the static implementation of HTML and CSS, and finally culminating in the dynamic integration of functionality using PHP, JavaScript, and database connectivity.

*Conceptualisation*

Before diving into the coding phase, we created a draft workflow document to map out how the webpages would interconnect (Fig. 2). This step provided a clear structure for the subsequent implementation and ensured a logical flow between the website's functionalities. Each page's purpose, input fields, and output expectations were detailed to serve as a blueprint for both the design and coding stages. This 'blueprint' aided us when connecting our final PHP pages together to ensure that we did not miss any connection between webpages and the database.

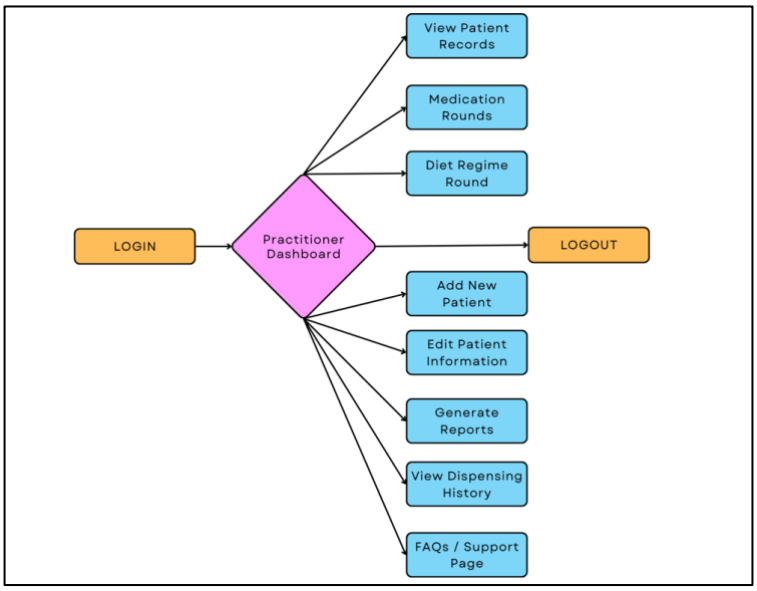


***Figure 2.*** *Initial draft workflow for website functionality.*

*Webpage Mock-ups*

The second step in the development process involved designing a mock version of the webpages using Canva (Fig 3). The goal was to visually conceptualise the first iteration of the website. Each team member provided input on the layout, colour scheme, and user experience elements. After reviewing multiple designs, we agreed on a cohesive theme that balanced simplicity with functionality. This step was essential in achieving team consensus and aligning the project's aesthetic direction. We wanted our design to be user-friendly and easy to use (easy navigation) but also contain several functionalities.

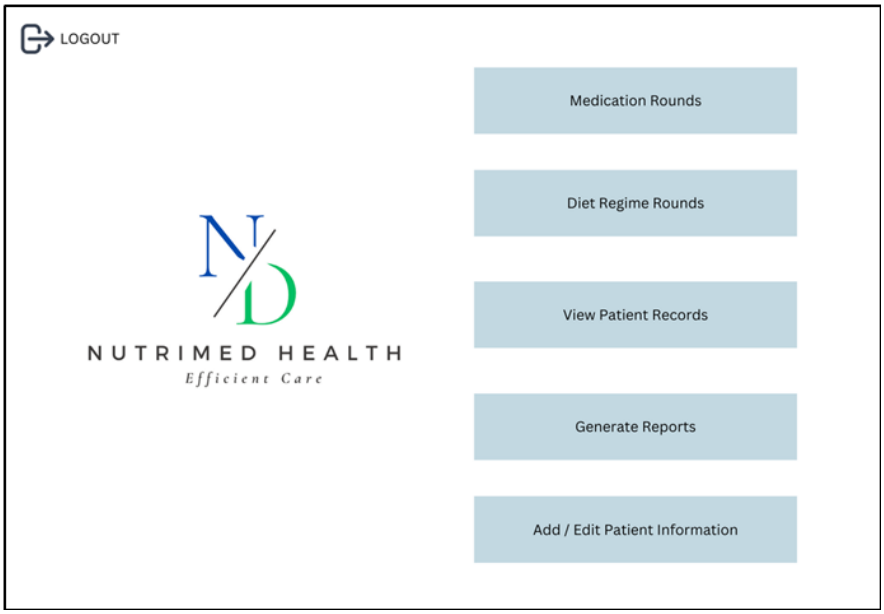


***Figure 3.*** *The initial design for our dashboard.*

Whilst it formed an easy link to our webpages, the design felt and looked simplistic, and we later updated the dashboard to incorporate more detail. The final NutriMed dashboard serves as a central hub, combining user-friendliness and functionality to support healthcare practitioners. It features a Key Contacts Section, which provides quick access to essential facility personnel, including pharmacy, security, IT support, and ward nurses. A photo gallery showcases patients enjoying activities, an Amenities List highlights the facility's services, and a Testimonials Section shows positive feedback from patients, offering a warm sense of a trusting community. Additionally, the Wellness and Support Section addresses practitioner well-being and professional development. It provides actionable wellness tips, alongside links to resources like time management guides, burnout prevention strategies, and medical journal databases.

For our 'Patient reports' section, we initially designed the interface to have a resident search bar on the left, and upon selecting a patient, a table showing medication and diet summaries would be displayed. For our final project we decide to have the option to form reports based on selection of either patient or practitioner. Rather than using the option to search for the patient, we incorporated a drop-down option to select patient or practitioner names. General styles of the webpage were updated according to our final colour schemes and CSS files.

In accordance with the project specifications, we decided that it would be optimal to display the medication and diet rounds for our patients by selecting the Day and the Round (morning, afternoon, evening). Our initial design choices on the overall medication and diet rounds are shown below (Figure 4). This general design was carried into our final design.

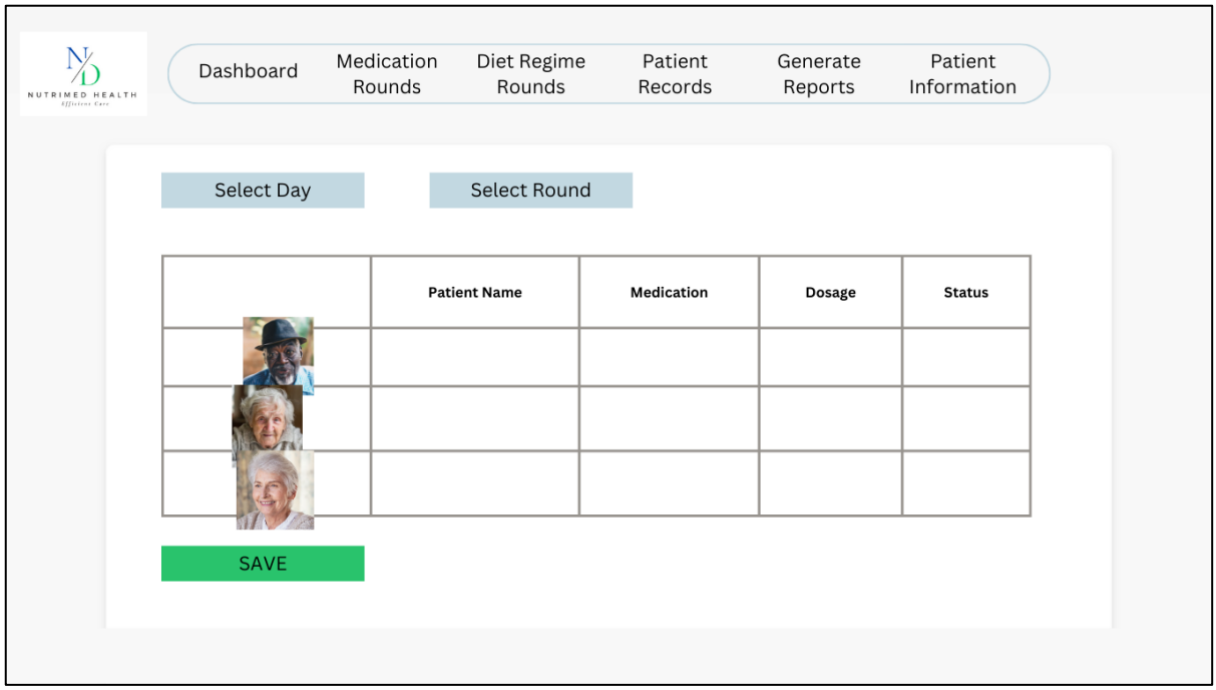


***Figure 4.*** *Initial designs for Medication Round section.*

Initial designs of patient records include a search bar feature to select the patient (on the left-hand side of the webpage), feature to prescribe medications, add new patient and edit current patients. Whilst the interface design of this page remained the same in the final design, we removed the 'prescribe new' and 'edit profile' functionalities on this page. We felt that the webpage appeared overly crowded, and some functionalities were added onto other pages; for instance, the ability to prescribe new patients was shifted to the 'manage orders' section.

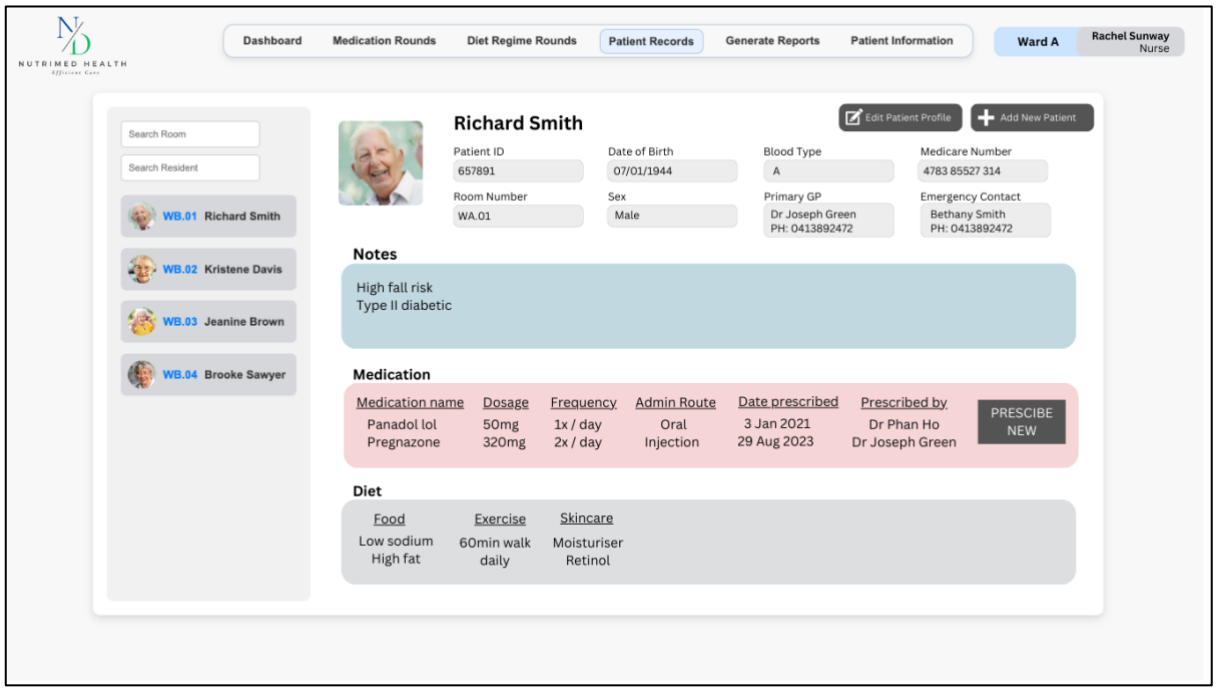


***Figure 5.*** *Initial design for Patient Records section.*

**HTML and CSS Implementation**

With the mock-ups finalised, the team transitioned to coding the website's structure using HTML and CSS. This stage laid the groundwork for the project, creating static versions of each webpage. Key elements, such as headers, navigation menus, and content areas were designed consistently across pages. CSS was used extensively to style the website, applying the colour palette and typography selected during the mock-up phase.

The first iteration of the website focused on establishing the foundational design and layout. The static pages served as a framework that could later be expanded with dynamic functionality. This version was presented to the team during a meeting, and the feedback gathered was incorporated into the design for refinement of the user experience.

The entire platform consistently uses a cool-toned color scheme (light blues, greens, and white gradients) to create a welcoming visual identity. Gradient backgrounds are used extensively across pages to add elegance and depth.

```
body {
    background: linear-gradient(180deg, #e8f9f0 0%, #ffffff 50%, #e8f9f0 100%);
    color: #333;
}
```

Tables are styled with alternating row colors and subtle shadows to improve readability and aesthetics. Incorporating hover effects on buttons, dropdowns, and images create an engaging and responsive experience for the user.

```
.right-column button:hover {
    background-color: #81bc9b; /* Pastel green */
    color: #ffffff;
}

.gallery-item:hover {
    transform: scale(1.05);
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.15);
}
```

**JavaScript Incorporation**

Once the static structure was in place, interactivity was added using JavaScript. This phase aimed to enhance the functionality of the website and improve the user experience. Form validation scripts were implemented to ensure that the data entered by the users met specified requirements, such as validating date formats, and required fields. Interactive navigation features, including dropdown menus and hover effects, were also added to make the website more dynamic.

JavaScript played a very important role in maintaining a seamless user experience, as it allowed for real-time feedback without requiring the page to reload. For example, errors in invalid form inputs appeared instantly, guiding users to correct any mistakes before submission. These

features collectively elevated the website from just a static platform to a more engaging and interactive tool.

For example, this script powers the report generation functionality of the platform, and allows used to create dynamic reports for specific patients or practitioners:

```
const patientMedRounds = medicalRounds.filter(round =>
   round.room_number === selectedRoom);
let reportContent = `
...
        <table>
          <thead>
            <tr>
              <th>Medication</th>
...
            </tr>
          </thead>
          <tbody>
            ${patientMedRounds.map(round => `
              <tr>
                <td>${round.medication_name}</td>
                <td>${round.practitioner_name}</td>
...
            `).join('')}
          </tbody>
        </table>
      ` : '<p>No medication records found</p>'}
...
`;
const pdfWindow = window.open('', '_blank');
pdfWindow.document.write(reportContent);
pdfWindow.document.close();
setTimeout(() => {pdfWindow.print();
}, 500);
```

This code filters the medical rounds based on the selected patient's room number to ensure that only relevant records are included in the report. Using template literals, it dynamically generates a well-structured report, including a table, title, and patient-specific data. It triggers the report to be opened in a new browser window, writes the HTML content and prompts the browser's print functionality, enabling the report to be printed or saved as a PDF.

Another example of an improved user experience is during the process of adding a new patient into the database. The text area for the patient notes auto-formats bullet points as the user types, providing a clean and structured input experience:

```
document.getElementById('notes').addEventListener('keydown', function(e) {
   if (e.key === 'Enter') {
      e.preventDefault();
      const start = this.selectionStart;
      const value = this.value;
```

```
        this.value = value.substring(0, start) + '\n• ' + value.substring(start);
        this.selectionStart = this.selectionEnd = start + 3;
    }
});
```

**Dynamic Content Integration – PHP**

To enable dynamic content integration, the static HTML files were subsequently converted into PHP files. This transition marked a turning point in the development process, as it allowed the website to interact with the database and deliver tailored content. Each webpage was integrated with PHP scripts to handle specific tasks, such as:

- Validating login credentials against the database
- Storing and retrieving patient information
- Dynamically generating reports for patients or practitioners

A script we used to setup the interaction with the database and PHP:

```
<?php
$servername = "localhost"; // MAMP default MySQL server
$username = "root";        // MAMP default username
$password = "root";        // MAMP default password
$dbname = "major_project_database"; // database name

// Create a connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

For example, the login page incorporated PHP scripts to validate user credentials with reference to the username and passwords in the database. If the input matched a record in the data, a session was initiated, granting the user access to otherwise restricted pages. Conversely, invalid credentials prompted an error message, ensuring only authorised users could log in. The patient information page was another highlight of this phase, as it allowed users to add new patient records easily via an HTML form connected to the database through PHP. Once users were successfully logged in, a session ID would be formed, and this would be used for downstream security, logout validation and processing the ‘Welcome’ message.

Another example is when the practitioner intends to insert a new record, the patient photos can be uploaded securely, with the script validating the file type and ensuring it is either JPEG, JPG, or PNG. Uploaded files are saved in a unique format to avoid overwriting existing files, and the directory is created if it doesn’t exist:

```
if (isset($_FILES['photo']) && $_FILES['photo']['error'] === UPLOAD_ERR_OK) {
    $allowedTypes = ['image/jpeg', 'image/png', 'image/jpg'];
    $fileType = $_FILES['photo']['type'];
```

```
    if (!in_array($fileType, $allowedTypes)) {
        $errors[] = 'Invalid file type. Only JPG, JPEG and PNG are allowed.';
    } else {
        $uploadDir = 'images/';
        if (!file_exists($uploadDir)) {
            mkdir($uploadDir, 0777, true);
        }
        $fileExtension = pathinfo($_FILES['photo']['name'], PATHINFO_EXTENSION);
        $uniqueFilename = 'profile_image_' . uniqid() . '.' . $fileExtension;
        $uploadPath = $uploadDir . $uniqueFilename;

        if (move_uploaded_file($_FILES['photo']['tmp_name'], $uploadPath)) {
            $photoPath = $uploadPath;
        }
    }
}
```

By the end of this phase, the website's core functionality was operational, with all pages successfully integrated with the backend database. Integration of PHP allowed us to connect our webpage and sure that if any forms were filled (e.g. the form to add a new patient to the aged-care facility), the results would be added to the files on our database. Furthermore, updates to the medications and diet rounds were connected to the database (using queries) so that once clicking 'save/update', the webpage would display update details (on the client side) and the database would also store these updates (on the server side). Server-side validation was seamlessly integrated with PHP.

**Session Management and Security Features**

As this is an internal platform that handles sensitive and patient information, security was a key focus during the website's development. PHP sessions were implemented to track user activity and restrict access to authenticated users. Upon logging in, the system produces a user ID (using cookie system) and ensures that only users with that specific ID have access to any of the software webpages (until logout). Within every webpage, a PHP script to configure session cookie parameters is added so that the login has been re-enforced (preventing access without login). For the logout page, a PHP script is set so that the session is destroyed, and the webpages cannot be accessed without re-login. By using the userID, the system presents a "Welcome [Name]" message displayed to personalise the user experience (the username that was placed when logging in is echoed).

Every webpage displays the "Welcome [Name]" button/container and upon clicking on this feature, a drop-down menu is opened to provide the user with the option to logout. Every page has a script that executes upon loading, to verify that the user is logged in. This was done using this code:

```
session_set_cookie_params([
    'secure' => false, // For local testing over HTTP
    'httponly' => true,
    'samesite' => 'Strict',
]);
```

```
session_start();

if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit();
}
```

To enhance security, practitioners cannot be added to the aged-care facility by signing-up from the login page, but rather they must be added by an administrator/staff that has login access and can enter the add new practitioner form. This ensures that users who simply sign up, won't be able to access the database and webpages.

**Form Validation**

Form validation ensures that on the client-side, users input data in the required fields and in the correct format, which thus ensures that user data is stored in the database in a consistent manner. If errors are present within form submission, feedback is immediately provided to the users, and the form can be re-submitted according to suggested guidelines. All the forms that are required by users to be submitted have been validated, including the form to add practitioners, form to add new patients, etc.

All required fields are validated, and these include first and last names, phone numbers (with digit limit set for the Australian audience), email validation, validation for sizing for pictures, date of birth, etc. Errors when submitting forms have been will show up in red and alert users of the correct methods to input the fields (e.g. if a phone number of 6 digits is placed, an error message will alert users that it needs to be 10 digits).

For instance, on the form to add a new patient, a date of birth needs to be provided (selected on calendar format), and this field has been PHP validated to ensure that the selected date of birth is not in the future, (Fig 6).

Code implemented:

```
// Date of Birth validation
    $dobDate = new DateTime($dob);
    $today = new DateTime();
    $minDate = clone $today;
    $minDate->modify('-120 years');

    if ($dobDate > $today) {
        $errors[] = 'Date of Birth cannot be in the future.';
    }

    if ($dobDate < $minDate) {
        $errors[] = 'Date of Birth cannot be more than 120 years ago.';
    }
```

All validations can be found embedded within the PHP files for the respective form (within webpage titles).

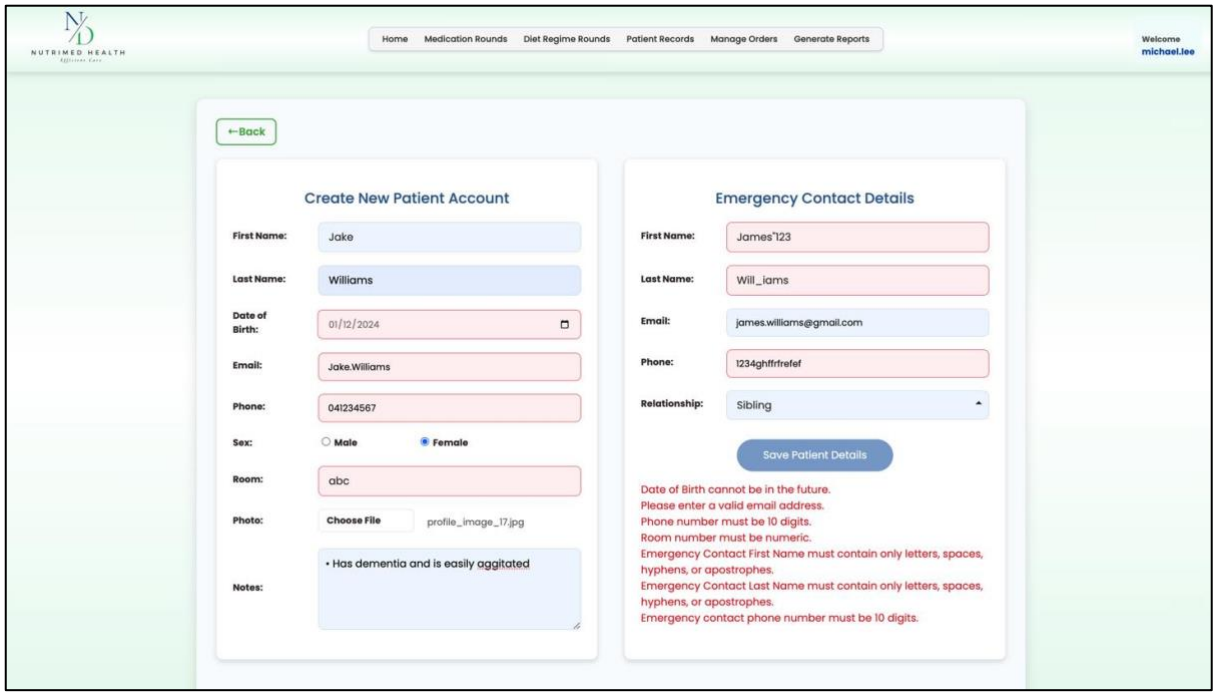


**Figure 6.** Form showing validation errors.

## Accessing the Project

To access this project, the webpage that should be accessed is the login.php file. The login.php file will open to the login page, where once the username and the password is entered, the user will be directed to the dashboard. Instructions on how to update and view the medications and diet regimes and preform all the functionalities of our database are outlined in the user manual. The usernames and passwords are located on the bottom of the last page of the user manual.