APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 1079-1098
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Soul Sports Physio and Rehab Booking Management System

Chee Seng Yik[1], Rabatul Aduni Sulaiman[1*]

[1] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: rabatul@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.058

Article Info

Received: 19 July 2025
Accepted: 20 November 2025
Available online: 30 November 2025

Keywords

Fitness Booking Management System, Mobile Application, Iterative Methodology

Abstract

Sports rehabilitation centers like Soul Sports Physio face operational inefficiencies from manual appointment, attendance, and data management, leading to scheduling conflicts and poor client experiences. This project addressed these challenges with the objectives to design a system using an object-oriented approach, develop it as a cross-platform mobile application using Flutter, and test its functionality and reliability. The resulting Soul Sports Physio and Rehab Booking Management System (SPRBMS) is a Flutter mobile app for clients and trainers and a web admin panel, using a Node.js backend and MongoDB database. Key features include Quick Response (QR) code session verification, real-time notifications, and payment processing. Functional and User Acceptance Testing confirmed SPRBMS streamlines bookings, improves data management, and offers a user-friendly interface, proving a viable digital transformation solution for specialized physiotherapy services.

1. Introduction

Sports rehabilitation and physiotherapy are essential in helping individuals recover from injuries, manage chronic conditions, and improve physical performance [1]. Centers like Soul Sports Physio in Kuala Lumpur provide these vital services but have historically relied on manual methods for client management, appointment scheduling, and progress tracking. These traditional processes, often involving paper records, spreadsheets, and communication via messaging apps like WhatsApp, are susceptible to human error, lead to disorganized appointment management, create difficulties in accurately verifying session attendance, and result in insufficient or error-prone reporting capabilities. This lack of an integrated digital system can significantly hinder operational efficiency and negatively impact on quality-of-service delivery. The problem statement identified a lack of accountability in training sessions due to difficulties in confirmation, challenges in managing appointments leading to double bookings, and insufficient reporting capabilities relying on manual Excel entry.

The primary motivation for this project was to address these identified inefficiencies at Soul Sports Physio. The main objectives set forth were to design a comprehensive Booking Management System using an object-oriented approach, to develop this system as a mobile application using Flutter technology for clients and trainers, supported by a web-based administrative panel, and to rigorously test the developed system to ensure its functionality, accuracy, and reliability through system testing and user acceptance testing.

The scope of the Soul Sports Physio and Rehab Booking Management System (SPRBMS) encompasses functionalities for three key user roles where clients allow booking appointments, viewing schedules, purchasing memberships, QR code generating for attendance, trainers allow managing appointments, inputting training volumes, scanning QR codes for session verification, and administrators allow managing user accounts, services, membership packages, generating summary reports, and overall system oversight. The system integrates features

such as real-time notifications and payment processing via Stripe. The expected outcome is a scalable, reliable, and user-friendly platform that enhances operational efficiency, reduces scheduling conflicts, improves client satisfaction, and ensures greater accountability in service delivery.

This paper contains five sections. Section 1 describes the project background. Section 2 reviews related work in the domain of booking and management systems and the technologies used. Section 3 details the methodology employed for the system's development. Section 4 presents and discusses the key results from the system implementation and testing phases. Finally, Section 5 offers a conclusion, summarizing the project's achievements and suggesting directions for future work.

2. Related Work

A Booking Management System is a software tool that helps schedule and manage appointments, reservations, or bookings for different services, making the process easier for both service providers and clients [2]. Soul Sports Physio and Rehab Centre previously operated with a manual process that led to inefficiencies. The technology stack for SPRBMS was chosen to address these issues. SPRBMS utilizes Flutter for mobile app development, Node.js and Express.js for its backend API, and MongoDB for secure data management. Real-time notifications are delivered via Socket.IO, QR codes track session attendance, and RESTful APIs facilitate scalable system communication. Dart, the language powering Flutter, supports advanced object-oriented programming concepts [3], and while it improves development efficiency [4], its community size is smaller than more established languages [5]. Flutter itself supports high-performance, cross-platform development [6], though its applications can be larger than native ones [7].

A comparative study of existing fitness center management systems was conducted to inform the design of SPRBMS. Peak Fitness is a fitness center in Malaysia offering members access to classes and personal training sessions via a web interface. Users can view schedules and select membership packages, but the system lacks a dedicated mobile application and advanced features like QR code verification or real-time appointment notifications [8]. Fitness First provides a comprehensive web platform for its members to book a wide range of group fitness classes across its various locations. While it has a robust scheduling system, it does not offer integrated features for personalized training reports or session attendance verification beyond simple check-ins [9]. Kamileon X is a mobile application for the Kamileon Fitness Centre, allowing members to book classes, manage appointments, and purchase class packages directly from their devices. It provides a convenient mobile-first experience but does not include specialized features like QR code verification for session authenticity or centralized training reports for administrative oversight [10]. Table 1 presents a comparative analysis of SPRBMS against these existing systems. While other systems offer basic scheduling, SPRBMS stands out with QR-code-based session verification and real-time notifications.

Table 1 *Comparison of the Existing System*

Features \ System	Peak Fitness	Fitness First	Kamileon X	SPRBMS (Proposed System)
Mobile App	×	×	✓	✓
Personal Training Session Booking	✓	✓	✓	✓
Package Purchase	✓	✓	✓	✓
Membership Management	✓	✓	✓	✓
QR Code Verification for Session	×	×	×	✓
Real-time Appointment Notifications	×	×	✓	✓
Client Points Tracking in Real-Time	×	×	×	✓
Payment Integration (Cards)	✓	✓	✓	✓
Sessions Timetable Viewing	✓	✓	✓	✓
Centralized Training Reports	×	×	×	✓

Based on Table 1, a comparative study of existing fitness center management systems such as Peak Fitness, Fitness First, and Kamileon X was conducted. While these systems offer functionalities like online class booking and membership management, they often lack specific features crucial for the nuanced needs of physiotherapy centers. For instance, none of the compared systems offered integrated QR code-based session verification for ensuring trainer and client presence, a key feature of SPRBMS designed to enhance accountability. Furthermore, SPRBMS provides a more tailored approach to training report management directly linked to client appointments and a mobile-first strategy for both client and trainer core interactions, differentiating it from more generic fitness platforms.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1079-1098

3. Methodology

The iterative model is a software development approach that emphasizes incremental improvement through repeated cycles, or iterations. Each iteration involves a complete development cycle, including planning, analysis, design, implementation, testing, review, and deployment. This approach enables the project to start with a basic version of the software and progressively expand its functionality by addressing issues and adding features identified in earlier cycles [11]. The iterative model was selected for the SPRBMS project due to its ability to detect defects early, as testing is conducted at the end of each iteration. This reduces risks and facilitates quick modifications before moving to subsequent iterations. Additionally, the model's division of the project into smaller components allows for efficient resource allocation and a steady workflow throughout the development lifecycle. Its flexibility also supports incorporating feedback from administrators, trainers, and clients, ensuring the system meets stakeholder needs. Each iteration involves adding new modules, testing, and refinement, continuing until the final system version is complete. Figure 1 illustrates the Software Development Life Cycle (SDLC) phase of this project using the Iterative Model.

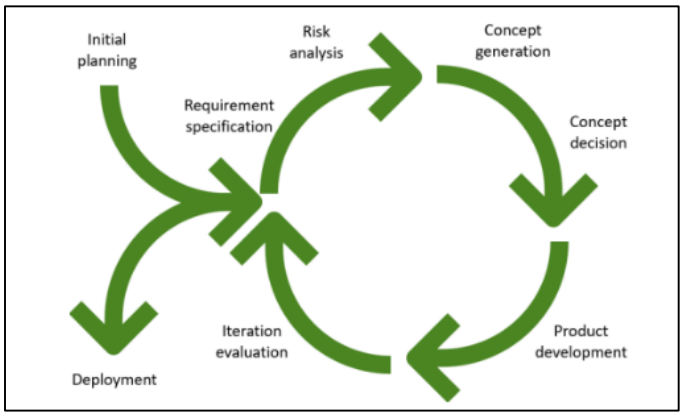


Fig. 1 *Iterative Model Phase [12]*

Each phase in Iterative Model includes specific tasks and deliverables, as summarized in Table 2. The workflow ensures each phase, from planning to deployment, is well-structured and contributes to the successful delivery of the SPRBMS.

Table 2 *System Development Work Flow*

Iteration	Phase	Task Description	Deliverables
Iteration 1	Planning	• Conduct stakeholder meetings • Define scope and objectives • Gather initial requirements • Create Gantt chart	Gantt chart, proposal
Iteration 1	Analysis	• Conduct interviews and observations • Study existing systems • Identify modules and system requirements	Requirement document
Iteration 1	Design	• Develop system architecture and UML diagrams • Design UI using Figma • Plan MongoDB schema	UML diagrams, prototypes
Iteration 1	Implementation	• Develop core modules (login, appointment booking) • Set up backend with Node.js & Express.js • Configure MongoDB and API endpoints	Functional modules
Iteration 1	Testing	• Conduct unit testing on core modules • Debug and validate features	Test results and overview

Table 2 *System Development Work Flow (cont)*

Iteration	Phase	Task Description	Deliverables

Iteration 1	Review	• Collect feedback from supervisor and stakeholders via Google Meet • Identify improvements	Evaluation notes
Iteration 1	Deployment	• Deploy prototype for limited internal use • Generate APK for demonstration	Prototype APK
Iteration 2	Analysis(Refined)	• Refine requirements based on feedback • Adjust and extend features	Updated requirements specification
Iteration 2	Design (Refined)	• Update UI/UX based on usability testing • Refine database schema	Updated UI designs, revised class diagrams
Iteration 2	Implementation	• Add modules: summary of training report, payment • Integrate real-time updates with Socket.IO	Additional system functionalities
Iteration 2	Testing	• Conduct system integration testing	Test case results
Iteration 2	Review	• Perform user acceptance testing (UAT)	UAT results
Iteration 2	Deployment	• Full app deployment • Hosting admin website	Mobile app APK file, Admin Panel Website

System requirements including functional requirements and non-functional requirements are presented in Table 3 and Table 4 respectively. The analysis highlights the specific functionalities, features, and qualities of the proposed system that align with the needs of stakeholders and end-users.

Table 3 *Functional Requirements*

No.	Module	Functionalities	User
1.	User Account Management	• The system shall allow Clients to register directly via the mobile app. • The system shall send an email verification link to the user's email address after account registration. • The system shall allow users to reset their password via a "Forgot Password" feature. • The system shall allow users to update and manage their profiles. • The system shall allow Admins to create new accounts and assign the role as Trainer or Client or Admin. • The system shall support role-based access control to limit functionalities based on user roles. • The system shall display user profile details. • The system shall allow Admins to create, update, and delete user accounts.	Admin, Trainer, Client
2.	Login and Logout	• The system shall allow users to log in and log out securely. • The system shall allow users to log in using their registered email and password. • The system shall verify email addresses before allowing access to the system. • The system shall validate user credentials during login.	Admin, Trainer, Client

Table 3 *Functional Requirements (cont)*

No.	Module	Functionalities	User

3.	Service Management	• The system shall allow Admins to add new services. • The system shall allow Admins to update or delete existing services. • The system shall allow Admins to display a list of available services to users.	Admin
4.	Training Report Management	• The system shall allow Trainers to submit and manage training reports. • The system shall allow Trainers to submit training reports after completing an appointment. • The system shall allow Admins to view submitted training reports. • The system shall store training report details for future reference.	Admin, Client
5.	Appointment Management	• The system shall allow Clients to view available time slots for a selected Trainer. • The system shall allow Clients to book appointments with a selected Trainer and input details such as date, time, and service type. • The system shall notify Trainers when a new appointment is made by a Client. • The system shall allow Trainers to confirm or reject an appointment request. • The system shall display appointments' status in both the Trainer's and Client's schedules. • The system shall allow Clients to generate a QR code for session attendance verification. • The system shall enable Trainers to scan the QR code to mark the session as completed. • The system shall ensure that completed appointments deduct 250 points from the Client's membership account. • The system shall prevent double bookings or overlapping appointments for Trainers. • The system shall allow Admins to monitor all appointments.	Admin, Trainer, Client
6.	Membership Management	• The system shall display a list of available membership packages to Clients. • The system shall allow Clients to purchase membership packages. • The system shall allow Admins to add new membership packages, including details. • The system shall allow Admins to update and delete existing membership packages. • The system shall track the membership status of each Client. • The system shall allow Admins to view membership purchase history.	Admin, Client

Table 3 *Functional Requirements (cont)*

No.	Module	Functionalities	User
7.	Payment Processing	• The system shall allow Clients to make payments for membership packages. • The system shall store payment details, including payment date, amount, and method. • The system shall allow Admins to view and manage payment histories. • The system shall notify Clients upon successful payment.	Admin, Client
8.	Report Generation	• The system shall allow Admins to generate reports related to training sessions and purchase records. • The system shall allow Admins to export reports in PDF format. • The system shall allow Admins to view detailed statistics on system usage.	Admin

Table 4 *Non-functional Requirements*

No	Modules	Functionalities
1.	Operational	• The system should be easy to use. • The system should have a responsive design, ensuring optimal user experience on different screen size devices. • The system shall ensure 24/7 availability for user login and profile management. • The system shall allow real-time updates of membership statuses after payment is processed. • The system should display all prices and handle payments in Ringgit Malaysia (MYR) as the default currency.
2.	Performance	• Login and user authentication shall be complete within 3 seconds. • Notifications for appointment updates shall be delivered to Clients and Trainers within 3 seconds of confirmation or rejection.
3.	Security	• The system should enforce password strength requirements (minimum 8 characters, including uppercase, numbers, and symbols). • The system shall integrate with a secure payment gateway to process membership payments. • Only Admins shall have access to create, update, or delete user accounts and system settings. • The system should deny access to any user if the username and password input are incorrect.
4.	Usability	• The system should provide a role-based dashboard with clearly labeled navigation for Admins, Trainers, and Clients. • All forms should provide clear error messages for invalid input.
5.	Integrity	• Membership points should be deducted accurately after each appointment. • Any updates to user profiles should be synchronized immediately. • Appointment status should always be accurate and updated in real-time.

A use case diagram provides a high-level of visual representation of what the system does and how the users are interacting with it. It also illustrates the relationship between the actors and the use cases, representing what the system can achieve. The use case diagram of the proposed system in Figure 2 highlights the roles of Admin, Trainer, and Client and illustrating their interactions with the various modules of the system in maintaining user accounts, login and logout, managing membership, managing services, managing appointments, managing training reports, processing payment, and generating reports. This diagram provides a basis for understanding the functional requirements of the system and the actors involved.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1079-1098

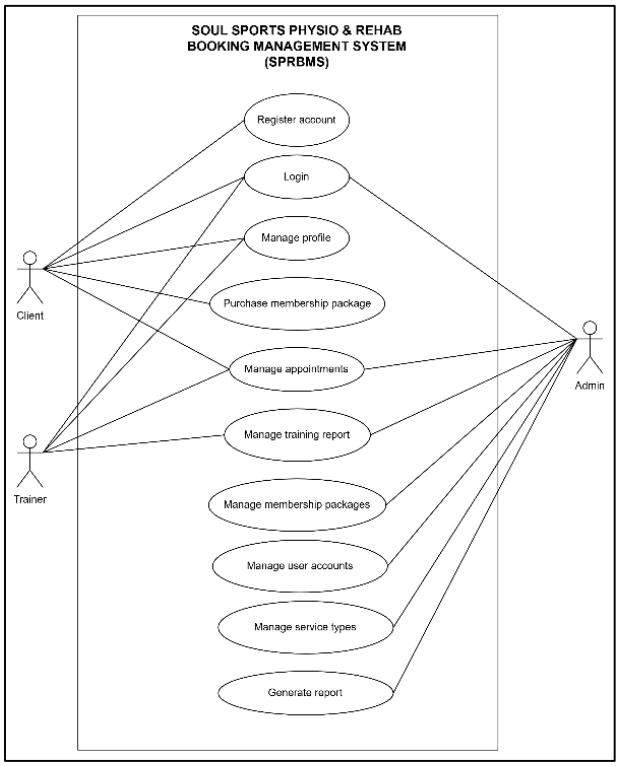


Fig. 2 *Use Case Diagram of SPRBMS*

The class diagram in Figure 3 represents a structured system for managing appointments, clients, trainers, and related activities. Each class is designed to handle specific aspects of the system's operations independently. There are a total ten classes, including Admin, Trainer, Client, Appointment, TimeSlot, MembershipPackage, ServiceType, PaymentRecord, TrainingReport, and TrainingSet .

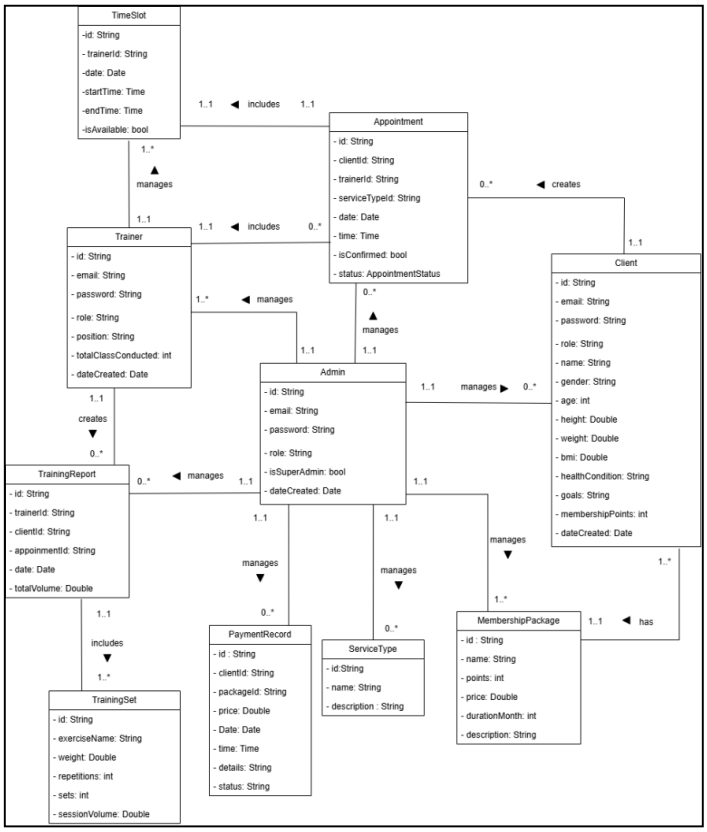


Fig. 3 *Class Diagram of the SPRBMS*

The database comprises collections for users, appointments, and reports, each designed to optimize data retrieval and storage. Table 5 lists the primary attributes of these collections.

Table 5 *Databases Collection*

Collection	Attributes
Admin	id, email, password, role, isSuperAdmin, dateCreated
Trainer	id, email, password, role, position, totalClassConducted, dateCreated
Client	id, email, password, role, name, gender, age, height, weight, bmi, healthCondition, goals, membershipPoints, dateCreated
TimeSlot	id,trainerId, date, startTime, endTime, isAvailable
Appointment	id, clientId, trainerId, serviceTypeId, date, time, isConfirmed, status
PaymentRecord	id, clientId, packageId, price, date, time, details, status
TrainingReport	id, trainerId, clientId, appointmentId, date, totalVolume
TraningSet	id, exerciseName, weight, repetitions, sets, sessionVolume
MembershipPackage	Id, name, points, price, durationMonth, description
ServiceType	id, name, description

User Interface design plays a crucial role in ensuring an intuitive and user-friendly experience for all users of the SPRBMS. Figure A.1 to A.11 in Appendix A illustrates a visual representation of user interface design for the proposed system. The developed system interface must meet the basic requirements such as it should be easy to use, user-friendly, and aligned with the specified functionalities. The user interface design aims to improve the interaction between system and users.

4. Results and Discussion

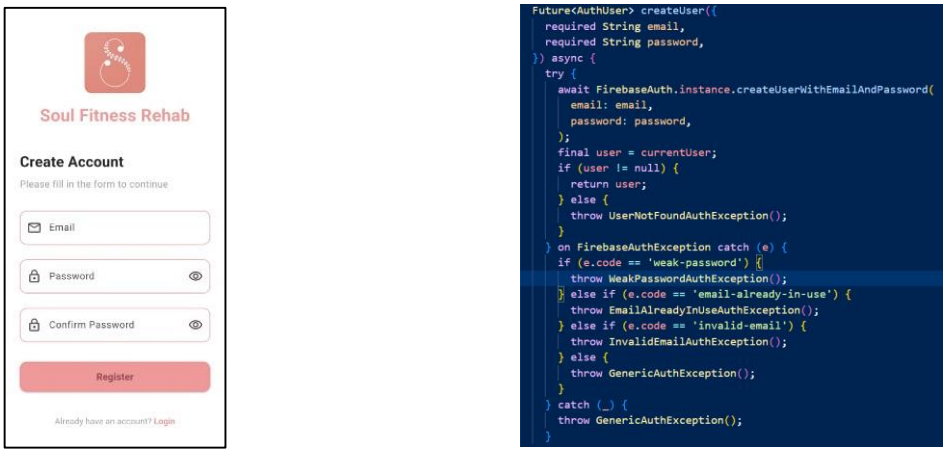
The implementation phase successfully translated the design specifications into a functional system with distinct interfaces and capabilities for each user role. The development of each module involved both frontend UI construction and backend logic, as detailed below with references to key interfaces and illustrative code segments.

4.1 Module Implementation

The implementation phase successfully translated the design specifications into a functional system with distinct interfaces and capabilities for each user role. The development of each module involved both frontend UI construction and backend logic, as detailed below with references to key interfaces and illustrative code segments.

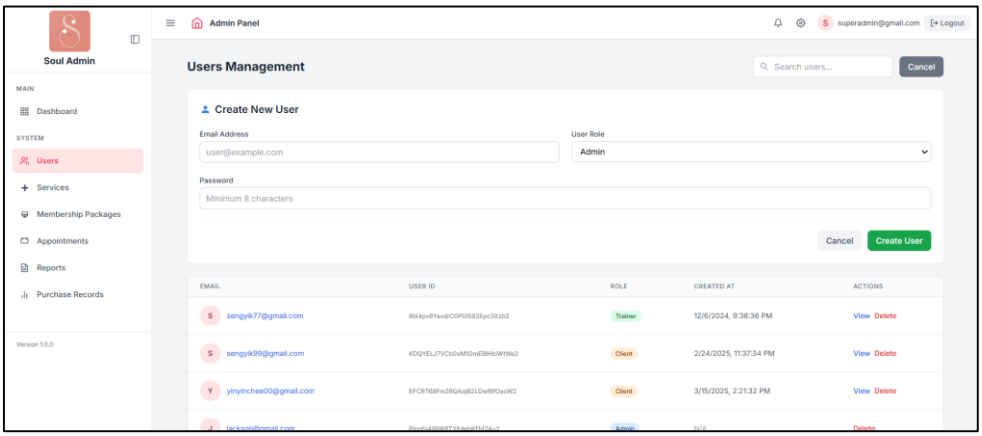
4.1.1 User Account Management Module

This module handles user registration and administration. Clients self-register via the mobile app interface as shown in Figure 4(a), providing essential details. The associated client-side code segment shown in Figure 4(b) demonstrates the use of Firebase Authentication's `createUserWithEmailAndPassword` method for new account creation and error handling for issues like weak passwords or existing emails. For administrators, a web panel as shown in Figure 5(a) allows for comprehensive user management. The backend code segments illustrate creating users with Node.js and Firebase Admin SDK, storing user data including roles in MongoDB as shown in Figure 5(b), deleting users from both Firebase Authentication and MongoDB as shown in Figure 5(b) and 5(c).



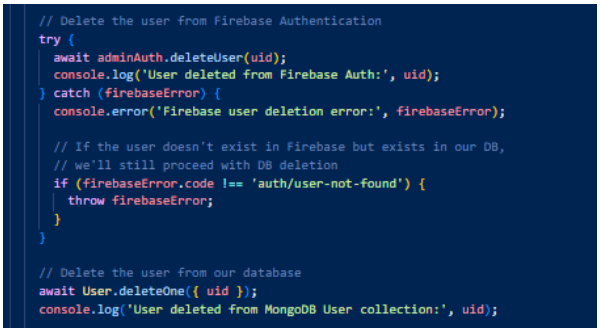
(a) (b)

Fig. 4 *Client Registerstion (a) Interface; (b) Code Segment*



(a)

(b)



(c)

Fig. 5 *Admin User Management (a) Interface; (b) Create User Code Segment; (c) Delete User Code Segment*

4.1.2 Login and Logout Module

Secure authentication was implemented for all roles. Clients and Trainers use a mobile login screen as shown in Figure 6(a), with client-side code as shown in Figure 6(b) utilizing Firebase Authentication's `signInWithEmailAndPassword` for verification. Logout for mobile users as shown in Figure 7(a) involves calling `FirebaseAuth.instance.signOut()` function to log user out from the app as shown in Figure 7(b).

(a) **(b)**

Fig. 6 *Client and Trainer Login (a) Interface; (b) Code Segment*

(a) **(b)**

Fig. 7 *Client and Trainer Logout (a) Interface; (b) Code Segment*

4.1.3 Training Report Management Module

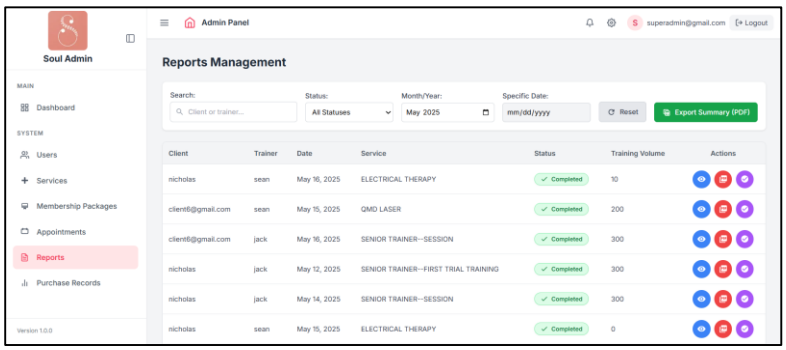
This module enables trainers to submit session report and administrators to review them. Trainers use a mobile interface to input exercise specifics and daily ratings as shown in Figure 8(a). The code for submitting reports in Figure 8(b) shows a report model creation and using provider to send HTTP POST request to a backend endpoint, sending structured report data including client and trainer IDs, appointment details, and performance metrics. Administrators manage and view all training reports via a web panel as shown in Figure 9(a), with backend logic to fetch and populate report data with associated client and trainer information as shown in Figure 9(b).

(a)

```dart
// Create report model
final report = ReportModel(
  clientUid: widget.appointment.client.clientUid,
  trainerUid: widget.appointment.trainer.trainerUid,
  appointmentId: widget.appointment.id,
  exercises: exercises,
  rpeDailyRating: double.tryParse(rpeController.text) ?? 0,
  sorenessDailyRating: double.tryParse(sorenessController.text) ?? 0,
  specialConsiderations: specialConsiderationsController.text,
  foodNotes: foodNotesController.text,
  waterNotes: waterNotesController.text,
  sleepNotes: sleepNotesController.text,
  reportStatus: status,
);

//submit report via report provider
ref.read(reportsProvider.notifier).submitReport(report).then((_) {
```

(b)

Fig. 8 *Trainer Submit Report (a) Interface; (b) Code Segment*



(a)

```javascript
const fetchReports = async () => {
  try {
    setLoading(true);
    const response = await fetch('/api/reports');
    if (!response.ok) {
      throw new Error('Failed to fetch reports');
    }
    const data = await response.json();
    setReports(data.reports || []);
  } catch (error) {
    console.error('Error fetching reports:', error);
    setError(error.message);
  } finally {
    setLoading(false);
  }
};
```

(b)

Fig. 9 *Admin Manage Training Report (a) Interface; (b) Code Segment*

4.1.4 Appointment Management Module

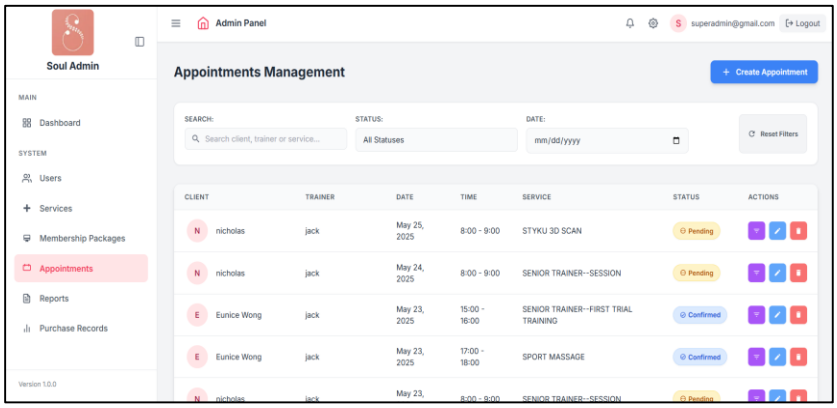
This central module handles the booking lifecycle. Clients book appointments via the mobile app as shown in Figure 10(a), and the corresponding code in Figure 10(b) sends booking data to a backend API. For session check-in, clients generate a unique QR code for their appointment as shown in Figure 11(a) and the code for this uses the `QrImageView` widget as shown in Figure 11(b). Trainers then scan this QR code using their mobile app's scanner to confirm attendance and complete sessions. Administrators oversee appointments through a web interface in Figure 12(a) with backend code supporting appointment creation while checking for scheduling conflicts as shown in Figure 12(b) and status updates in Figure 12(c).

(a) (b)

Fig. 10 *Client Book Appointment (a) Interface; (b) Code Segment*

(a) (b)

Fig. 11 *Client Check In (a) Interface; (b) Code Segment*



(a)

```
// Check for scheduling conflicts
const appointmentDate = new Date(body.appointmentDate);
const conflictQuery = {
  trainerId: body.trainerId,
  appointmentDate: {
    $gte: new Date(appointmentDate.setHours(0, 0, 0, 0)),
    $lt: new Date(appointmentDate.setHours(23, 59, 59, 999))
  },
  status: { $in: ['confirmed', 'pending'] },
  $or: [
    {
      'appointmentTime.start': { $lt: body.appointmentTime.end },
      'appointmentTime.end': { $gt: body.appointmentTime.start }
    }
  ]
};

const existingAppointment = await Appointment.findOne(conflictQuery);

if (existingAppointment) {
  return NextResponse.json(
    { error: 'Scheduling conflict with an existing appointment' },
    { status: 409 }
  );
}

// Create the appointment
const newAppointment = await Appointment.create(body);
```

(b)

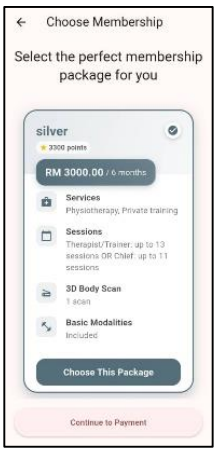
```
// Check if appointment exists
const appointment = await Appointment.findById(id);

if (!appointment) {
  return NextResponse.json(
    { error: 'Appointment not found' },
    { status: 404 }
  );
}

// Update only the status field
appointment.status = body.status;
await appointment.save();

return NextResponse.json({
  message: 'Appointment status updated successfully',
  appointment: appointment
});
} catch (error) {
console.error('Error updating appointment status:', error);
return NextResponse.json(
  { error: 'Failed to update appointment status' },
  { status: 500 }
);
}
```

(c)

Fig. 12 *Admin Manage Appointment (a) Interface; (b) Create Appointment Code Segment; (c) Update Appointment Status Code Segment*

4.1.5 Membership Management Module

This module facilitates client subscriptions and administrative control over packages. Clients select and view membership packages on the mobile app as shown in Figure 13(a), with code fetching available packages from the backend as shown in Figure 13(b). Administrators manage these packages as shown in Figure 14(a) through a web panel. The backend code supports creating new membership packages in Figure 14(b) and updating existing ones in Figure 14(c).

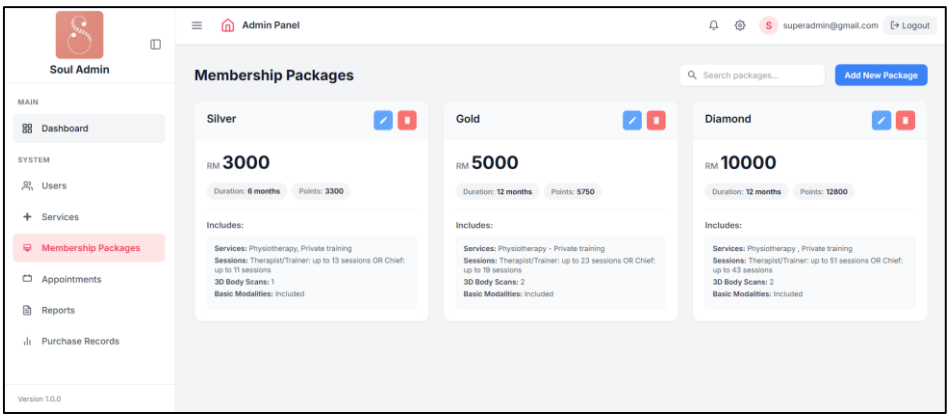


(a)

```
Future<List<MembershipPackage>> getMembershipPackages() async {
  String? token = await FirebaseAuthProvider().fetchToken();
  const String apiUrl = BackendConfig.apiUrl;
  final response = await http.post(
    Uri.parse('${apiUrl}membership-packages'),
    headers: {
      'Content-Type': 'application/json',
      'Authorization' : 'Bearer $token'
    },
    );

  try {
    if (response.statusCode == 200) {
      List<dynamic> data = json.decode(response.body);
      List<MembershipPackage> membershipPackages = data.map((json) => MembershipPackage.fromJson(json)).toList();
      return membershipPackages;
    } else {
      throw Exception('Failed to load membership packages');
    }
  } catch (e) {

    throw Exception('Failed to load membership packages $e.toString()');
  }
```

(b)

Fig. 13 *Client Select Membership Package (a) Interface; (b) Code Segment*



(a)

```
// Create new package
const newPackage = await MembershipPackage.create(body);

return NextResponse.json(
  { message: 'Membership package created successfully', package: newPackage },
  { status: 201 }
);
} catch (error) {
console.error('Error creating membership package:', error);
return NextResponse.json(
  { error: 'Failed to create membership package' },
  { status: 500 }
);
}
```

(b)

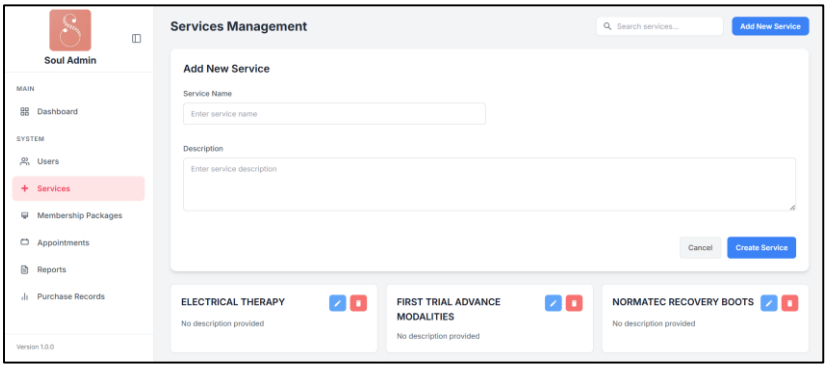
```
// Update the package
const updatedPackage = await MembershipPackage.findByIdAndUpdate(
  id,
  body,
  { new: true, runValidators: true }
);

return NextResponse.json({
  message: 'Membership package updated successfully',
  package: updatedPackage
});
catch (error) {
console.error('Error updating membership package:', error);
return NextResponse.json(
  { error: 'Failed to update membership package' },
  { status: 500 }
);
```

(c)

Fig. 14 *Admin Manage Membership Package (a) Interface; (b) Create Code Segment; (c) Update Code Segment*

4.1.6 Service Management Module

Administrators define and manage the services offered by Soul Sports Physio. A web interface as shown in Figure 15(a) allows them to add new services and view existing ones. The backend logic includes code for creating new service types with validation as shown in Figure 15(b) and updating service details while checking for name conflicts as shown in Figure 15(c).



(a)

```
// Check if service with the same name already exists
const existingService = await ServiceType.findOne({ name: body.name });
if (existingService) {
  return NextResponse.json(
    { error: 'A service with this name already exists' },
    { status: 409 }
  );
}

// Create new service
const newService = await ServiceType.create(body);
```

(b)

```
// Check if service exists
const existingService = await ServiceType.findById(id);
if (!existingService) {
  return NextResponse.json(
    { error: 'Service not found' },
    { status: 404 }
  );
}

// Check for name conflict (but ignore the current service)
const nameConflict = await ServiceType.findOne({
  name: body.name,
  _id: { $ne: id }
});

if (nameConflict) {
  return NextResponse.json(
    { error: 'Another service with this name already exists' },
    { status: 409 }
  );
}

// Update the service
const updatedService = await ServiceType.findByIdAndUpdate(
  id,
  body,
  { new: true, runValidators: true }
);
```

(c)

Fig. 15 *Admin Manage Service (a) Interface; (b) Create Code Segment; (c) Update Code Segment*

4.1.7 Payment Processing Module

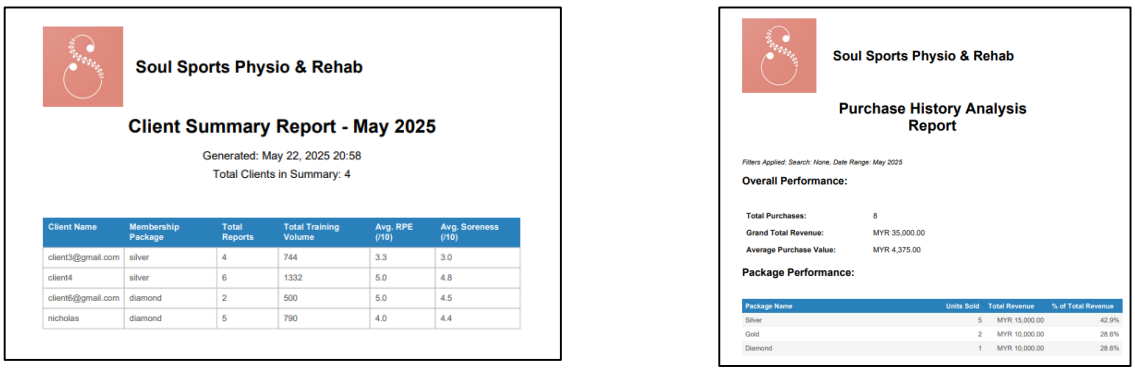
Secure payments for membership packages are handled by integrating the Stripe API. The client-side mobile interface as shown in Figure 16(a) presents a Stripe payment sheet where users can enter card details. The associated code segment in Figure 16(b) demonstrates creating a payment intent on the backend, initializing the Stripe payment sheet with the client secret, and presenting the payment sheet to the user for transaction completion.

(a) **(b)**

Fig. 16 *Client Payment (a) Interface; (b) Code Segment*

4.1.8 Report Generation Module

This module provides administrators with reporting capabilities. They can generate summary PDF reports for client training activities as shown in Figure 17(a) and purchase history analyses in Figure 17(b). The code segments for generating these PDFs illustrated in Figure 18(a) and Figure 18(b) utilizes libraries like jsPDF to structure and format data fetched from backend APIs into downloadable documents.



(a) **(b)**

Fig. 17 *Summary Report (a) Training Report; (b) Purchase History Report*

(a) **(b)**

Fig. 18 *Code Segment (a) Training Report; (b) Purchase History Report*

4.2 System Testing

System testing was systematically conducted to validate that all modules of the SPRBMS operate in accordance with their specified requirements. A comprehensive suite of test cases was executed across eight main system modules. As summarized in Table 6, a total of 48 test cases were executed, all of which passed successfully, indicating a high degree of functional reliability for the implemented system.

Table 6 *Overall Result of Functional Test Cases*

Module	Total Test Cases	Total Success	Total Failed
User Account Management	10	10	0
Login/Logout	5	5	0
Training Report Management	3	3	0
Appointment Management	16	16	0
Membership Management	6	6	0
Service Management	3	3	0
Payment Processing	3	3	0
Report Generation	2	2	0
Total	**48**	**48**	**0**

4.3 User Acceptance Testing

User Acceptance Testing (UAT) was conducted to validate the SPRBMS against real-world scenarios and end-user expectations. The testing involved 10 respondents, comprising a mix of the system's intended user roles including clients, trainers, and administrators, who were asked to perform a series of predefined tasks. Feedback was systematically collected via a structured Google Form. The questionnaire was divided into three main sections such as System Functionality, Ease of Use and Learnability, and System Interface and Visual Design, utilizing a 5-point Likert scale for quantitative feedback, from 1-Strongly Disagree to 5-Strongly Agree. An additional open-ended question gathered qualitative suggestions. The results are discussed below.

Respondents expressed high satisfaction with the system's overall functionality. Most users strongly agreed or agreed that the SPRBMS functions well and effectively meets user requirements as shown in Figure 19. Key feedback highlighted the system's success in reducing data entry errors and making the booking processes at Soul Sports Physio seamless. Furthermore, users strongly felt that the system would enhance their task effectiveness and integrate well with their typical workflows.

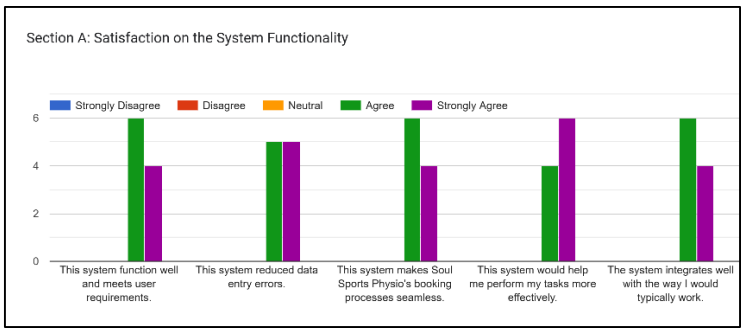


Fig. 19 *Users' Satisfaction on System Functionality*

The SPRBMS was perceived as highly user-friendly and easy to learn. Most participants indicated that it was simple to learn how to use the system. There was strong agreement regarding the clarity of information organization on the screens and the logical, straightforward nature of the steps required to complete tasks. On-screen messages and prompts were also generally found to be clear and helpful, contributing to a positive learnability experience. The result from the question regarding the ease of use and learnability shown in Figure 20.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1079-1098

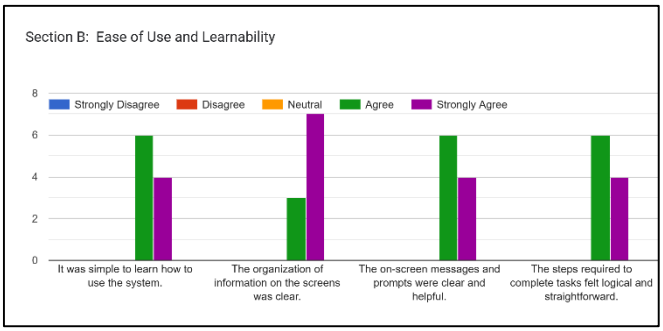


Fig. 20 *Ease of Use and Learnability*

The visual design and interface of the SPRBMS were well received. A significant number of users agreed or strongly agreed that the system's aesthetics such as colors, fonts, and layout are appealing as shown in Figure 21. There was a strong consensus that information is presented in a clear, readable manner, and that the use of icons and buttons is consistent and understandable. The overall interface was described as modern and professional.

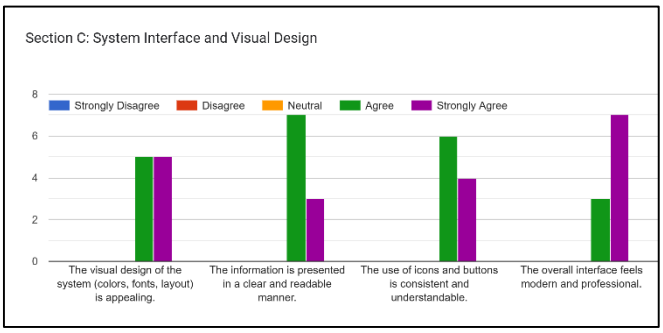


Fig 21 *System Interface and Visual Design*

5. Conclusion

This project successfully concluded with the Soul Sports Physio and Rehab Booking Management System (SPRBMS) achieving its objectives of designing, developing, and rigorously testing a comprehensive digital solution to replace inefficient manual processes at Soul Sports Physio. The implemented system, featuring streamlined appointment booking, enhanced accountability via QR-code session verification, and improved data management, was validated through functional testing and received strong positive feedback during User Acceptance Testing regarding its functionality, usability, and modern design. While SPRBMS offers significant operational advantages, key recommendations for future work include diversifying payment methods, enhancing new user onboarding, and optimizing performance for extensive historical data, positioning the system as a successful application of technology in specialized service delivery with clear pathways for continued improvement.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia (UTHM) and for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] N. Jesingh, P. Dhankher, R. Batra Malik, and D. Choudhry, "Athletes' expectations about physiotherapy in sports injury rehabilitation: A literature review of qualitative studies," *International Journal For*

Multidisciplinary Research, 2023. [Online]. Available: https://pdfs.semanticscholar.org/30a0/dacf6b50b67f7278a5fbb8ef4c8dfa425f3e.pdf.

[2] K. Merritt and S. Zhao, "Software design and development of an appointment booking system: A design study," in *Proc. EAI Int. Conf. Body Area Networks*, Cham, Switzerland, Dec. 2021, pp. 275–294, doi: 10.1007/978-3-030-95593-9_21.

[3] A. M. Hassan, "JAVA and DART programming languages: Conceptual comparison," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 17, no. 2, pp. 845–849, Feb. 2020. [Online]. Available: https://www.semanticscholar.org/reader/4aa90271fcb9625127f1aa3c280ecee1d40a35ea.

[4] R. Vindua, D. Handayani, and A. Ekrinifda, "Implementation of Dart programming language in mobile-based DRs snack sales application design," *J. Comput. Netw. Archit. High Perform. Comput.*, vol. 6, no. 3, pp. 1080–1088, 2024. [Online]. Available: https://www.researchgate.net/publication/382190353_Implementation_of_Dart_Programming_Language_in_Mobile-Based_DRs_Snack_Sales_Application_Design

[5] A. Badkar, "What is Dart programming? Everything you need to get started!," *Simplilearn.com,* Jul. 24, 2024. [Online]. Available: https://www.simplilearn.com/what-is-dart-programming-article. [Accessed: Jul. 17, 2025].

[6] W. Wu, "React Native vs Flutter, cross-platforms mobile application frameworks," M.S. thesis, Metropolia Univ. of Applied Sciences, Helsinki, Finland, 2018. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf.

[7] D. Palumbo, "The Flutter framework: Analysis in a mobile enterprise environment," Ph.D. dissertation, Dept. of Control and Computer Eng., Politecnico di Torino, Turin, Italy, 2021. [Online]. Available: https://webthesis.biblio.polito.it/secure/19111/1/tesi.pdf.

[8] Peak Fitness. (n.d.). *Peak Fitness: Aspire. Achieve. Ascend*. [Online]. Available: https://peakfitness.com.my/. [Accessed: Jul. 17, 2025].

[9] Fitness First Malaysia. (n.d.). *Fitness First: Elevate your fitness*. [Online]. Available: https://www.fitnessfirst.com/my/en.

[10] Kamileon Fitness. (n.d.). Kamileon Fitness. [Mobile application]. Available: Google Play Store.

[11] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile software development: Methodologies and trends," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 7, pp. 246–270, 2020. [Online]. Available: https://online-journals.org/index.php/i-jim/article/view/13269/7405.

[12] K. J. Blakstad and O. Tingsborg, "The iterative development method: Its aspects and effect on innovation within new product development," M.S. thesis, KTH Royal Inst. of Technol., Stockholm, Sweden, 2023. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1792859/FULLTEXT01.pdf.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1079-1098

Appendix A: Interface Design of SPRBMS

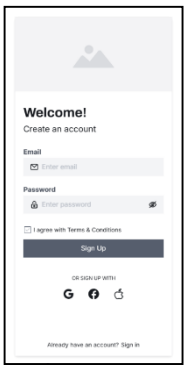


Fig. A.1 Register Account *Interface*

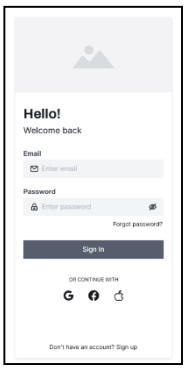


Fig. A.2 *Login Interface*

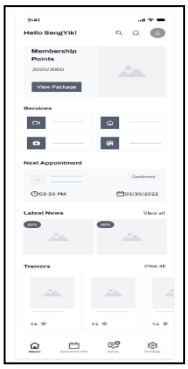


Fig. A.3 *Home Screen Interface*

Fig. A.4 *Manage Appointment Interface*

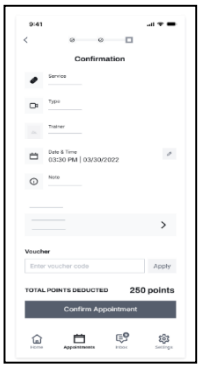


Fig. A.5 *Appointment Confirmation Interface*

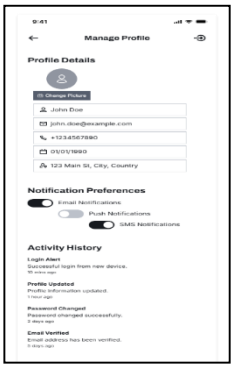


Fig. A.6 *Manage Profile Interface*

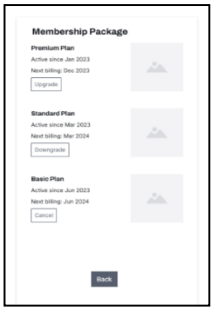


Fig. A.7 *Purchase Membership Package Interface*

Fig. A.8 *Manage Training Report Interface*

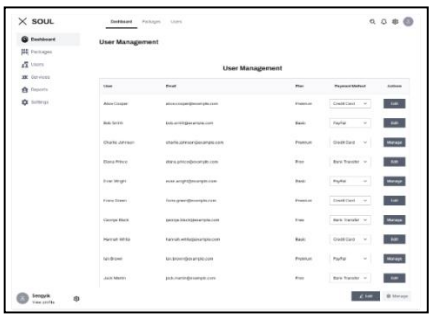


Fig. A.9 *Manage User Interface*

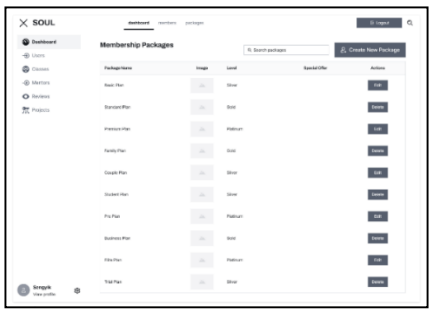


Fig. A.10 *Manage Membership Package Interface*

Fig. A.11 *Generate Report Interface*