Gym Management System

(ZFit)
Information Technology Project (IT2080)

Group ID:

ITP25_B2_W225

Campus : Malabe
Group Number : ITP25_B2_W225

	IT Number	Student Name	Student E-mail Address	Contact Number
1	IT 23 5621 10	H R G N Karunathilaka	IT23562110@my.sliit.lk	0764891241
2	IT 23 5418 70	N M B M L B Nawaratne	IT23541870@my.sliit.lk	0763541666
3	IT 23 5804 80	K B P Kavisika	IT23580480@my.sliit.lk	0718827129
4	IT 23 7536 62	O A l De Silva	IT23753662@my.sliit.lk	0740357412
5	IT 23 7414 78	B Dhayabari	IT23741478@my.sliit.lk	0770772330

Repo: https://github.com/gaindunuhansith/ZFit

Contents

1.Progress

1.User , attendance , membership management – H R G N Karunathilaka – IT 23 5621 10

The ZFit system is based on the User Management module where managers, staff, and users are managed and granted entry. New users register using personal details and confirm their accounts using email, and current users identify themselves through role-based access to data. The information is written by members, passwords are reset by members, staff view a little, managers gain full administrative access and all this takes less than one second and has a user friendly interface across all devices.

The Attendance module makes it easy to track members using the QR code technology that enables a member to check-in and out with ease in any ZFit facility. Patients simply scan QR codes generated by the facilities in order to record visits in real time with automatic membership verification, and staff create session-specific QR codes. The system captures check-in time and duration, and usage data, providing both members and managers with attendance and usage insights respectively and requiring less than two seconds to run with encrypted security and mobile access.

Membership Management module deals with the complete cycle of subscriptions, and admits a variety of plans on different terms and on different levels of access to the facilities. Members follow their status and renewal dates using automatic reminders, build plans, activate and follow expirations with detailed audit trails. Coupled with attendance to confirm access and retention information, it offers secure and responsive operations; and queries are executed instantly and on all devices.

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a member, I want to register an account and log in with JWT, sessions and cookies authentication so that I can access gym services securely.	Yes	**KARUNATHILAKA H R G N - IT23562110**
As a member, I want to view and update my profile information so that my details are current	Yes	

As a member, I want to purchase and view my membership details so that I can track my gym access.	Yes	**KARUNATHILAKA H R G N - IT23562110**
As a member, I want to check in/out using my unique QR code so that I can access the gym quickly and track my visits.	Yes	
As a member, I want to view my attendance history so that I can monitor my gym usage patterns.	Yes	
As a member, I want to see which facilities are included in my membership plan so that I know what services I can access	Yes	
As a member, I want to renew, cancel and pause my current membership status.	No, Partially implemented	

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a staff member, I want to scan QR codes and validate memberships at entry so that only authorized members access the facility.	Yes	**KARUNATHILAKA H R G N - IT23562110**
As a staff member I want too member information to better serve the members	Yes	
As staff I want to checkin using qr code and see my attendance history.	Yes	
As a staff, I want to create, update, cancel, pause, and extend memberships so that I can manage member subscriptions effectively.	Yes	
As a staff member, I want to manually enter attendance records and force check-in for special cases so that I can handle exceptions appropriately.	No	

High level user story/function	Completed (Y/N, Comment)	Responsible member (Name)
As a manager, I want to view membership subscriptions and filter by status so that I can proactively manage membership subscriptions.	Yes	**KARUNATHILAKA H R G N - IT23562110**
As a manager/staff, I want to see searchable and filterable membership plans, members, attendance.	Yes	
As a manager, I want to generate custom reports about users, members, memberships.	Yes	
As a manager, system it should automatically generate unique qr code for each user so I can better manage attendance and access control.	Yes	

High level user story/function	Completed (Y/N, Comment)	Responsible member (Name)
As a member, staff member, or manager, I want to reset my password securely via email so that I can regain access to my account if I forget my credentials	Yes	
As a system, I must enforce roles — managers (full CRUD operations), staff (read only), members (access to the store) — so that permissions are clear.	No, Partially implemented	

2.Inventory Control – N M B M L B Nawaratne – IT 23 5418 70

The Inventory Control module is the place where the gym keeps track of everything it owns and sells. It covers two kinds of items, day to day gym equipment (like treadmills and dumbbells) and saleable products (like supplements) that members can buy. Managers use this module to add new items, update details, and remove old or damaged items. Staff who are authorized can look up items and see their status quantities, and notes, but only managers are allowed to make changes.

The system makes it easy to see what you have what condition it is in and what needs attention. If an equipment item is broken or being serviced, managers can mark it as 'Under Maintenance' and record what was done and when. For saleable products, the module watches stock levels and warns you when quantities are low, so you can reorder before you run out. You can also download clear, ready to share reports to review inventory, low stock items, and maintenance history.

Everything is designed to be fast and reliable. Searches and updates are built to complete in under two seconds even with more than a thousand items. Access is secure, only authorized users can make changes, and every important action is recorded. Whether you are on a desktop, laptop, or phone, the interface is simple to use helping the team keep shelves stocked, equipment safe, and operations running smoothly.

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a member, I want to browse Sellable items and purchase them, so that I can buy supplements or anything from the gym.	Yes	**N M B M L B Nawaratne IT 23 5418 70**
As a member, I want to view my order history. So that can track past purchases.	No	

High level user story/function	Completed (Y/N, Comment)	Responsible member (Name)
As a manager/staff, I want to see searchable and filterable categories, items and suppliers in the inventory, to find categories, items, suppliers quickly.	Yes	**N M B M L B Nawaratne IT 23 5418 70**
As a manager/staff, I want to see the stock details in the inventory so that I can get idea about stock levels	Yes	
As a manager/staff, I want to see the transaction history so that I can audit activity.	No	
As a manager/staff, I want a visual badge in the items and stock tables for low stock items. So that I can spot issues at a glance	Yes	

High level user story/function	Completed (Y/N, Comment)	Responsible member (Name)
As a manager, I want to add any category to the system, so that it appears in inventory.	Yes	**N M B M L B Nawaratne IT 23 5418 70**
As a manager, I want to create items with required type (Saleable or Equipment), so that fields and workflows match the item's purpose.	Yes	
As a manager, I want to create a supplier to the system, so that I can onboard vendors.	Yes	
As a manager, I want to update inventory details to keep current data.	Yes	
As a manager, I want to remove any inventory record, so that obsolete assets don't clutter active inventory.	Yes	
As a manager, I want to link a default supplier to an item, so that reorders can be initiated faster.	Yes	

As a manager, I want to set a low stock level for the sellable items, so that the system knows when to trigger low stock.	Yes	**N M B M L B Nawaratne IT 23 5418 70**
As a manager, I want an email when the item hit their low stock level, so that I can reorder item.	No	
As a manager, I want to change the equipment status to under maintenance, so that I can take it out tracked until fixed.	Yes	
As a manager, I want to record maintenance completion date, so that maintenance history and schedules stay accurate.	Yes	
As a manager, I want to download all the inventory reports, so that I can make decisions and share updates.	Yes	

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a system, I want to check item quantity against low stock level on every stock change, so that low stock is caught promptly.	Yes	**N M B M L B Nawaratne IT 23 5418 70**
As a system, I want to decrement the stock level after successful purchase, so that inventory stays accurate.	Yes	
As a system, I must enforce roles, managers (full CRUD operations), staff (read only), members (access to the store), so that permissions are clear.	No	

3.Payment Control- K B P Kavisika -IT 23 5804 80

This is because the Payment module is the full-scale financial processing unit of ZFit and it will take into consideration various payment options such as PayHere online payments, bank transfers with the receipt upload, cash payments, and others. It can process different kinds of payments, like

membership, inventory, booking, and general, and it is well validated to be safe to use. The PayHere integration allows effortless online payments with webhook support that automatically reserves the payment status and bank transfers that demand uploading receipts and passing an approval procedure through an administrator. Every payment is carefully monitored using special transaction identities, gateway responses, and failure reasons, which helps in issuing partial refunds and ensuring good audit trails to remain financially compliant.

The payment system is fully administratively controlled by managers, and has specific interfaces where they can review pending bank transfers, approve or reject payments with comments, and keep statistics on payment. It is an automatic system that creates professional invoices with a unique numbering system (ZF-INV-YYYY-XXXXXX) which contains an itemized breakdown with tax calculation and the due date. The financial records are stored securely using role-based access controls and are encrypted in order to prevent anyone other than an authorized person to edit the financial records. Customers and staff are informed on the status of a payment in real-time and via email notifications across the payment lifecycle.

The payment system was designed to be reliable and high-performing, taking less than three seconds to complete and automatically reconcil and clear stale pending payment. It provides detailed error management, Zod-schemas to validate the input, and offline fallbacks. The responsive design is compatible with all devices, and there are streamlined dashboards to view payment history, create reports, and handle refunds. All financial operations are recorded with a time stamp, establishing a history that cannot be changed or adjusted by accounting and conflicts.

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a member, I want to make secure card payments for memberships and orders.	Yes	**KBP KAVISIKA IT 23 5804 80**
As a member, I want to via bank transfer by uploading receipts image, so I can pay when cards aren’t available.	Yes	
As a member, I want to view my personal payment history, so I can track my spending.	Yes	
As a member, I want to request refunds for my payments, so I can get my money back for unnecessary payments.	Yes	
As a member, I want real time status updates(email) during the payment process,	No	

so that I can immediately notify if my payment success or failed.		**KBP KAVISIKA IT 23 5804 80**
As a member, I want multiple payment methods, so I can pay the method I want when according to the situation.	Yes	

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a system, I want to automatically generate invoices when payments are completed without manual intervention.	Yes	**KBP KAVISIKA IT 23 5804 80**
As a system, I want to automatically clean up expired pending payments after 10mins, so I can keep database clean, and performance is maintained.	Yes	
As a system, I want to verify PayHere webhook signatures using MD5 hash validation, so that only legitimate payment notifications are processed.	Yes	
As a system, I want to handle different payment types (membership, inventory, booking), so I can match business logics.	Yes	
As a system, I want to send automated email notifications for payments and refund s confirmations or failures, so users will immediately inform.	No	
As a system, I want to maintain detailed audit logs of all payment-related activities, so that I can handle compliance and troubleshooting requirements.	Yes	

High level user story/function	**Completed (Y/N, Comment)**	**Responsible member (Name)**
As a manager, I want to create manual invoices for custom billings, so that I can	Yes	

handle special cases and flexible billing requirements.		**KBP KAVISIKA IT 23 5804 80**
As a manager, I want to manage bank transfer approvals with receipts verification, so I can handle offline payments properly before the service activation.	Yes	
As a manager, I want to review and approve or decline the refund requests, so I can maintain refund decisions and provide feedbacks from members.	Yes	
As a manager, I want to view comprehensive payment history with filtering options, so I can track all financial transactions and generate reports.	Yes	
As a manager, I want to generate Excel payment reports for accosting and analysis, so financial data can export and shared.	Yes	
As a manager, I want role role-based access payment features (full access for manager, limited for staff),so that financial data security is maintained.	Yes	

4. Booking control - B DHAYABARI- IT 23 7414 78

Our system provides a comprehensive Booking Management module that ensures both gym members and administrators can efficiently handle class reservations. Members can easily browse available classes, select a suitable trainer and facility, and reserve their preferred time slots. During booking, the system automatically links selected names (class, trainer, facility) with their respective IDs to avoid errors and ensure data accuracy.

Members are allowed to book, cancel, or reschedule their reservations, but only before the booking date, ensuring fair usage of available seats. The system enforces strict validation rules such as class seat limits (maximum 20), minimum duration (40 minutes), and fee restrictions to guarantee proper data entry.

Every booking is securely stored in the database and connected to the member's profile, allowing users to track their past and upcoming reservations. A calendar view is integrated into the dashboard, where members can see their own bookings by date, while administrators and managers can view all members' bookings.

Administrators and managers have full control over booking records, including the ability to add, edit, or delete reservations. They can also generate detailed reports on a daily, weekly, monthly, or yearly basis, which provide insights into booking trends with information such as booking ID, member name, class, trainer, facility, booking type, date, time, and price.

High Level User Story	**Completed(Y/N), Comments**	**Responsible member**
As a member, I want to browse available classes, trainers, and facilities, so that I can choose the right booking for me.	Yes	IT23741478 B DHAYABARI
As a member, I want to book a class by selecting class, trainer, and facility, so that my booking is linked correctly.	No	
As a member, I want to cancel or reschedule my booking before the booking date, so that I don't lose flexibility.	yes	
As a member, I want to view my bookings in a calendar format, so that I can quickly check my schedule.	No	

As a member and admin, I want to see error messages when I enter invalid data (e.g., 0 seats, short duration), so that I know how to fix my booking.	yes	
As an admin/manager, I want to manage (add/edit/delete) all bookings, so that I can maintain data accuracy	yes	
As an admin/manager, I want to generate reports (daily/weekly/monthly/yearly), so that I can analyze booking trends.	No	
As an admin/manager, I want to enforce seat, duration, fee, trainer experience, and facility equipment validations, so that no invalid booking data enters the system.	yes	

5.User Progress - O.A.I. De Silva - IT 23 7536 62

Our fitness platform ZFit includes a fully functional Function Tracking and Progress Management module designed to help members and trainers monitor fitness goals efficiently. This module allows members to record, update, and visualize their workout and nutrition progress, while managers and admins can supervise and analyze member performance trends.

Each member can log daily workouts (e.g., cardio, weight training) and nutritional details (e.g., calories, protein intake). Progress data is automatically converted into visual graphs and performance summaries on the user dashboard. Trainers (managers) can view these logs, give feedback, and generate progress reports to ensure members stay consistent with their fitness routines.

Admins can oversee the entire system, verify data accuracy, and generate analytical reports for evaluation purposes. The system also provides data validation, access control (user-only access for updates), and report generation functionalities integrated with the backend API and MongoDB database.

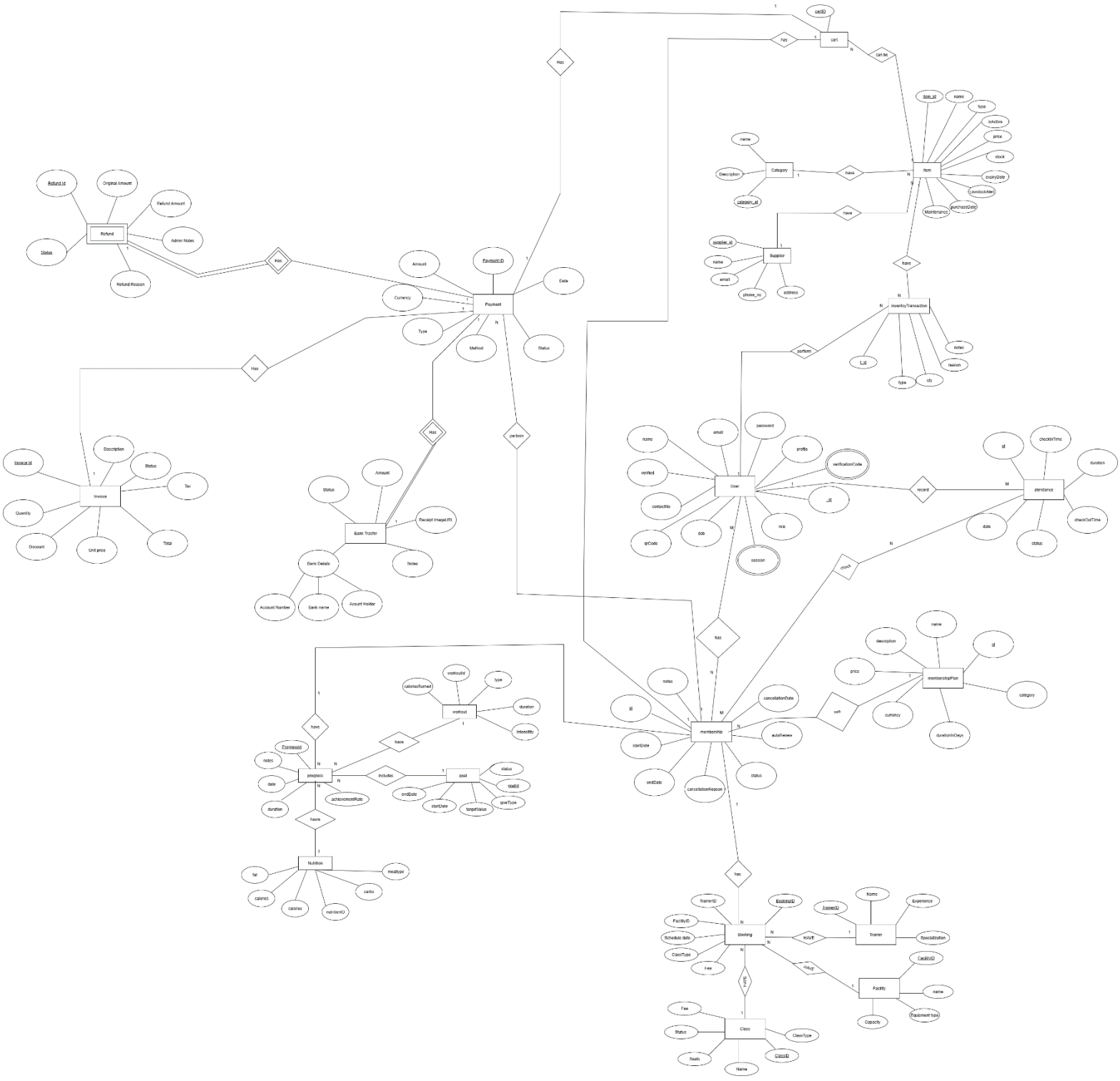
This feature improves user engagement and provides clear progress visibility to both users and trainers, ensuring that fitness goals are continuously tracked, analyzed, and optimized.

High level user story/ function	Completed (Y/N, comment)	Responsible member (Name)
As an admin, I want to view all members' fitness progress records, so that I can monitor their improvements.	Yes	**O.A.I. De Silva IT 23 7536 62**
As an admin, I want to generate progress reports for members, so that I can track performance over time.	Yes	
As a manager, I want to view each member's workout and nutrition progress, so that I can guide them better.	Yes	
As a manager, I want to update or correct progress data, so that all information stays accurate.	Yes	
As a manager, I want to delete incorrect or duplicate progress entries, so that the database remains clean.	Yes	
As a member, I want to record my daily workout progress, so that I can track my fitness goals.	Yes	
As a member, I want to record my nutrition intake, so that I can monitor my diet plan.	Yes	

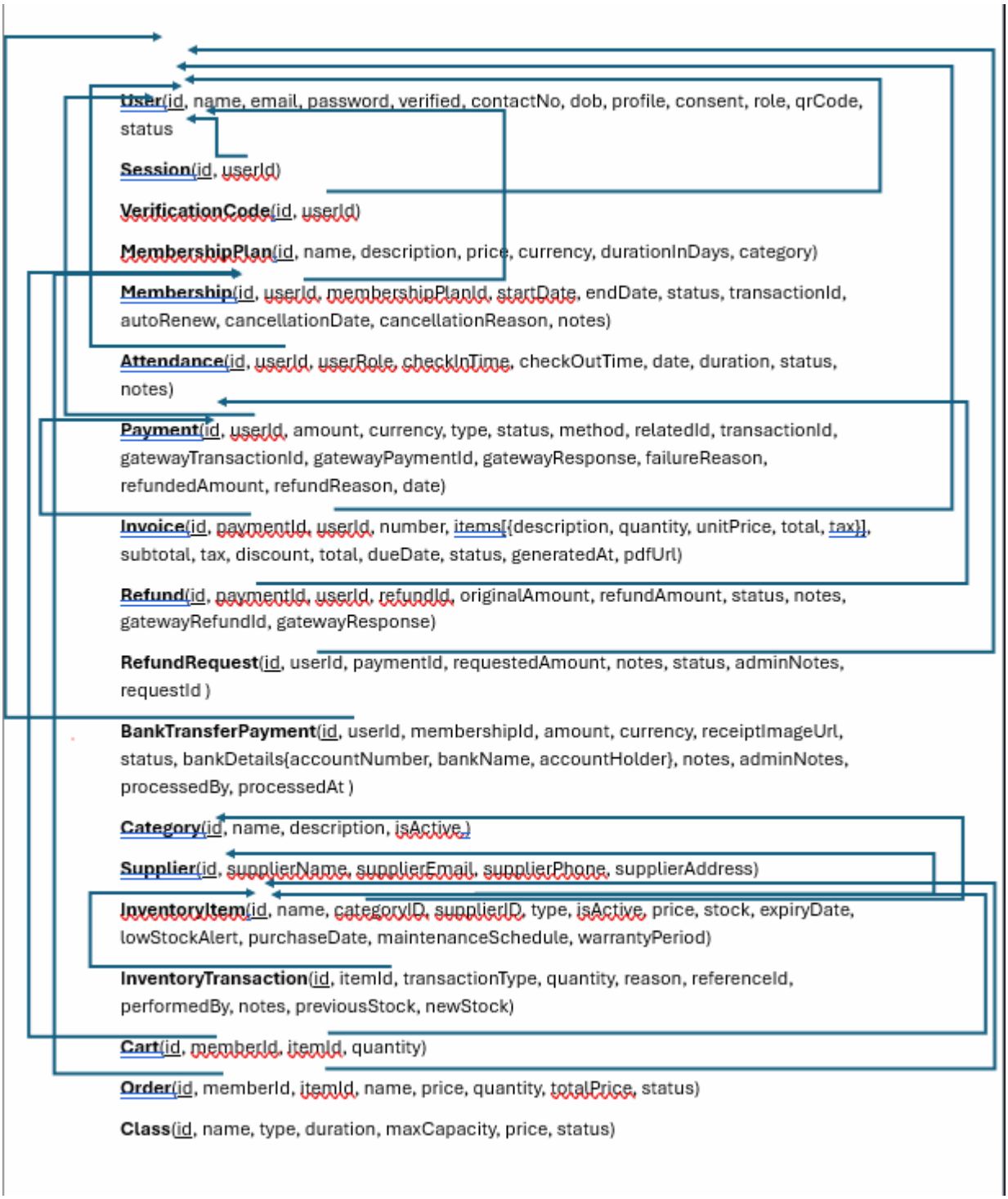
As a member, I want to view graphs of my workout and nutrition progress, so that I can visualize my achievements.	Yes	
As a member, I want to edit or update my previous progress entries, so that I can fix any mistakes.	Yes	
As a member, I want to receive progress summaries or milestones, so that I stay motivated.	Yes	
As an admin, I want to view all members' fitness progress records, so that I can monitor their improvements.	Yes	
As a member, I want to set new fitness goals (e.g., weight loss or strength gain), so that I can personalize my progress tracking.	Yes	
As a member, I want to filter and view my progress by week, month, or goal type, so that I can analyze trends more clearly.	Yes	**O.A.I. De Silva** **IT 23 7536 62**
As a member, I want to download my progress report, so that I can share it with my trainer or keep it for my records.	Yes	
As a manager, I want to delete incorrect or duplicate progress entries, so that the database remains clean.	Yes	
As a member, I want my progress data to stay private and only visible to me and my trainer, so that I feel secure about my information.	No	
As a member, I want to view comparisons between my planned goals and actual progress, so that I can evaluate my consistency.	No	

2.ER Diagram

ER diagram Link - [Zfit-ER-Diagram]



3.Normalized Schema



Trainer(id, name, specialization, status)

Booking(id, memberId, classId, trainerId, facilityId, scheduledDate, cancellationDeadline, status) Fk user.id

Goal(id, memberId, goalType, target, deadline, assignedBy) Fk user.id

Progress(id, memberId, workoutsCompleted, attendance, goalsAchieved, date) Fk user.id

FitnessGoal(id, memberId, goalType, target, deadline, status) Fk user.id

Nutrition(id, memberId, mealType, calories, protein, carbs, fats, notes, date) Fk user.id

Workout(id, memberId, exercise, sets, reps, weight, notes, date) Fk user.id

WorkoutLog(id, memberId, date, workout, duration, notes) Fk user.id

WorkoutSimple(id, memberId, type, duration, caloriesBurned, date) Fk user.id

4.Test Case Design

Test Case Designed by: Karunathilaka H R G N

Test Case Designed Date: October 4, 2025

Test Case ID: UM001
Testing Function: User registration and profile management
Test Priority: High
Module Name: User Management
Dependencies: Valid email service, database connection, frontend registration form
Test Steps: 1. Navigate to user registration page 2. Fill in all required fields (name, email, phone, password, role) 3. Accept GDPR and marketing consent 4. Submit registration form 5. Verify email verification process 6. Login with new credentials

7. Update user profile information 8. Verify profile changes are saved

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
UM01	Name: "John Doe", Email: "john@example.com", Phone: "+94712345678", Password: "ValidPass123!", Role: "member", GDPR: true	User account created successfully, verification email sent	User created, email sent	Pass
UM02	Name: "", Email: "invalid-email", Phone: "123", Password: "weak"	Validation errors for all invalid fields	Form shows specific validation errors	Pass
UM03	Valid registration data, but duplicate email	Error: "Email already exists"	Duplicate email error displayed	Pass
UM04	Valid user login, then update profile with new address	Profile updated successfully	Profile changes saved	Pass

Test Case 2: Membership Management - Membership Creation and Status Updates

Test Case Designed by: Karunathilaka H R G N

Test Case Designed Date: October 4, 2025

Test Case ID: MM001
Testing Function: Membership lifecycle management
Test Priority: High
Module Name: Membership Management
Dependencies: Valid user account, membership plan exists, admin/staff login
Test Steps: 1. Login as admin/staff user 2. Navigate to membership management section 3. Create new membership for existing user 4. Select membership plan and set dates

5. Submit membership creation
6. Update membership status (pause/resume)
7. Cancel membership with reason
8. Verify membership history and status changes

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
MM01	User ID: valid_user_id, Plan ID: premium_plan, Start Date: today, Auto-renew: true	Membership created successfully with active status	Membership created, status: active	Pass
MM02	Valid membership ID, Status: "paused", Reason: "Medical leave"	Membership paused successfully	Status updated to paused	Pass
MM03	Valid membership ID, Status: "cancelled", Reason: "Moving abroad"	Membership cancelled with reason recorded	Status: cancelled, reason saved	Pass
MM04	Invalid user ID, Valid plan ID	Error: "User not found"	User validation error	Pass

Test Case 3: Membership Plans Management - Plan Creation and Modification

Test Case Designed by: Karunathilaka H R G N

Test Case Designed Date: October 4, 2025

Test Case ID: MP001

Testing Function: Membership plan CRUD operations

Test Priority: Medium

Module Name: Membership Plans Management

Dependencies: Admin/manager login, database access

Test Steps:

1. Login as admin/manager
2. Navigate to membership plans section
3. Create new membership plan with all details
4. Set pricing, duration, and category
5. Save the membership plan
6. Edit existing plan details

7. Update price and duration 8. Verify plan is active and available for selection

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
MP01	Name: "Premium Plan", Price: 5000, Duration: 30, Category: "weights", Currency: "LKR"	Membership plan created successfully	Plan saved with all details	Pass
MP02	Name: "Premium Plan" (duplicate), Price: 3000, Duration: 15, Category: "yoga"	Error: "Plan name already exists"	Duplicate name validation error	Pass
MP03	Valid plan ID, Update price: 6000, Duration: 45	Plan updated successfully	Price and duration changed	Pass
MP04	Name: "", Price: -100, Duration: 0	Validation errors for all invalid fields	Form shows validation errors	Pass

Test Case 4: Attendance Management - QR Code Check-in/Check-out System

Test Case Designed by: Karunathilaka H R G N

Test Case Designed Date: October 4, 2025

Test Case ID: AM001
Testing Function: QR code based attendance tracking with membership validation
Test Priority: Medium
Module Name: Attendance Management
Dependencies: Valid user with active membership, camera access, QR scanner functionality
Test Steps: 1. Login as staff/admin user 2. Navigate to attendance scanning page 3. Grant camera permissions 4. Scan valid member's QR code 5. Verify membership validation

6. Process check-in/check-out
7. Verify attendance record creation
8. Test with expired membership

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
AM01	Valid QR code from user with active membership	Check-in successful, attendance record created	User checked in, record saved	Pass
AM02	Valid QR code from user with expired membership	Error: "No active membership found. Access denied."	Membership validation error	Pass
AM03	Invalid QR code format	Error: "Invalid QR token"	QR validation error	Pass
AM04	Same user scans QR twice (check-in then check-out)	Firstscan: check-in, Second scan: check-out	Status toggled correctly	Pass

Test case Designed by: Nawaratne N M B M L B

Test case Designed Date: October 2, 2025

Test Case ID:	TC-INV-001
Testing Function:	CRUD operations for inventory items & categories
Test Priority:	High
Module Name:	Inventory management – Gym equipment & supplements
Dependencies:	Database(InventoryItem, Category, Supplier tables)

Test steps:

1. Open Add Item form, enter valid details → verify record inserted into DB.
2. Add Item with missing required field (Name empty) → verify validation error shown.
3. Update an existing Item's stock count → verify DB record updated.
4. Delete Item by ID → verify record removed.

5. Add Category → verify entry saved with status = "Active".

Test Case ID	Input	Expected Output	Actual Output	Status
TC-INV-001	Add item (valid details: Protein Powder, Category: Supplements, Type: Sellable	Item record inserted into DB with status = "Active"	Item record created successfully and visible in inventory list	Pass
TC-INV-002	Add Item (missing Name)	Validation error "Item Name is required"	System displayed validation error message	Pass
TC-INV-003	Update Item Stock Quantity	Item record updated in DB	Quantity updated in database and reflected in inventory	Pass
TC-INV-004	Delete Item by ID	Item removed from DB	Item still visible in inventory due to soft delete bug	Fail

Test Case ID:	TC-INV-005
Testing Function:	CRUD operations for Equipment management
Test Priority:	High
Module Name:	Inventory management – Equipment Tracking
Dependencies:	Database(InventoryItem, Supplier tables)

Test steps:

1. Open Add Equipment form, enter valid details → verify equipment record created.
2. Add Equipment with missing Purchase Date → verify validation error.
3. Update Equipment maintenance date → verify DB record updated.
4. Search items by category → verify filtered results displayed.
5. Generate Equipment report → verify report contains correct data.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-INV-005	Add Equipment (Treadmill, Type: equipment, Purchase Date: 2025-01-15)	Equipment record inserted with maintenance tracking enabled	Equipment successfully added with maintenance schedule	Pass
TC-INV-006	Add Equipment (missing Purchase Date)	Validation error "Purchase Date is required for equipment"	System displayed validation error message	Pass
TC-INV-007	Update Equipment Maintenance Date	Equipment maintenance date record updated in DB	Maintenance date update	Pass
TC-INV-008	Search Items by Category name Equipment	Filtered list showing only gym equipment	Search returned correct filtered results	pass

Test Case ID:	TC-INV-009
Testing Function:	Stock Management & Low Stock Alerts
Test Priority:	High
Module Name:	Inventory management – Stock control
Dependencies:	Database (InventoryItem, InventoryTransaction tables), Email Service

Test steps:

1. Update sellable item stock below minimum threshold → verify low stock alert triggered.
2. Process sale transaction → verify stock automatically decremented.
3. Add stock via purchase transaction → verify stock incremented.
4. Generate stock report → verify accurate stock levels displayed.
5. Test email notification to managers → verify emails sent successfully.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-INV-009	Update Protein Powder stock to 5 (below minimum 10)	Low stock alert email sent to all managers	Email notifications sent successfully to 4 managers	Pass
TC-INV-010	Process sale of 3 Protein Powder units	Stock decremented by 3, transaction recorded	Stock updated correctly, transaction logged	Pass
TC-INV-011	Add purchase of 50 Protein Powder units	Stock incremented by 50, purchase transaction recorded	Stock updated successfully	Pass
TC-INV-012	Generate Stock Report	Report showing all the stock levels in the inventory right now	Report generated with accurate low stock items	pass

Test Case ID:	TC-INV- 013
Testing Function:	Shopping Cart & Member Orders
Test Priority:	Medium
Module Name:	Inventory Management - Member Shopping
Dependencies:	Database (Cart, InventoryItem, Member tables)

Test steps:

1. Member adds item to cart → verify cart record created/updated.
2. Update cart item quantity → verify cart updated in DB.
3. Remove item from cart → verify item removed from cart.
4. Process cart checkout → verify inventory stock decremented.

5. View cart contents → verify accurate item list and totals.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-INV-013	Member adds Whey Protein (Qty: 2) to cart	Cart record created/updated with item and quantity	Cart successfully updated for member	Pass
TC-INV-014	Update cart quantity from 2 to 5	Cart quantity updated to 5 in database	Quantity updated successfully	Pass
TC-INV-015	Remove item from cart	Item removed from cart record	Item successfully removed from cart	Pass
TC-INV-016	Process cart checkout	Inventory stock decremented, order created	Stock updated, order processed successfully	pass

Test Case ID:	TC-INV- 017
Testing Function:	Supplier Management & Purchase Orders
Test Priority:	Medium
Module Name:	Inventory Management - Supplier Operations
Dependencies:	Database (Supplier, InventoryItem tables)

Test steps:

1. Add new supplier with valid details → verify supplier record created.
2. Add supplier with duplicate email → verify validation error.
3. Update supplier contact information → verify DB record updated.
4. View items by supplier → verify filtered supplier inventory.
5. Generate supplier report → verify accurate supplier data.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-INV-017	Add Supplier (NutriSupply, email: info@nutrisupply.com)	Supplier record inserted into database	Supplier successfully added and available for selection	Pass
TC-INV-018	Add Supplier with duplicate email	Validation error "Email already exists"	System displayed duplicate email error	Pass
TC-INV-019	Update Supplier phone number	Supplier contact updated in DB	Phone number updated successfully	Pass
TC-INV-020	Filter items by supplier "NutriSupply"	Display all items from selected supplier	Correct supplier items displayed	pass

Test Case Designed by: Kavisika K B P
Test Case Designed Date: October 4, 2025

Test Case ID: PP001
Testing Function: Processing payments through PayHere gateway
Test Priority: High
Module Name: Payment Management
Dependencies: User must be logged in, PayHere gateway integration
Test Steps: 9. Login as a registered user 10. Navigate to the membership/booking payment page select any. 11. Select payment method and enter card details. 12. Submit payment request to PayHere 13. Verify payment status update 14. Transaction receipt will be sent to the given email.

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
PP01	Amount:500 Type: Membership Method: Card	Payment record created with status 'completed'	Payment created successfully	Pass
PP02	Amount: -100 Type: Membership Method: Card	Error: Amount must be positive	System displays validation error	Fail
PP03	Invalid user(unregistered) Amount:5000 Type: Membership Method: Card	Error: Invalid user Id	Payment failed	Fail

Test Case Designed by: Kavisika K B P
Test Case Designed Date: October 4, 2025

Test Case ID: PR001
Testing Function: Processing refund requests
Test Priority: Medium
Module Name: Refund Management
Dependencies: Completed payment record, refund policy
Test Steps: 1. Select a completed payment for refund 2. Enter refund amount and reason 3. Process refund through PayHere 4. Update payment and create refund record

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
PR01	Payment.Amount:5000 Refund.Amount:5000 Reason: User cancellation	Full refund processed, status=’completed’	Refund completed successfully	Pass
PR02	Payment.Amount:5000 Refund.Amount:2500 Reason: Partial refund	Partial refund processed, refund Amount=2500	Partial refund completed	Pass
PR03	Payment.Amount:5000 Refund.Amount:6000 Reason: Over refund	Error: Refund amount exceeds payments amount	System declined over refund	Fail
PR04	Already refunded payment Refund.Amount:6000	Error: Payment already refunded	Error displayed correctly	Fail

Test Case Designed by: Kavisika K B P
Test Case Designed Date: October 4, 2025

Test Case ID: BT001
Testing Function: Bank transfer slip upload and verification
Test Priority: Medium
Module Name: Bank Transfer Management
Dependencies: User account, bank transfer record
Test Steps: 1. User uploads bank transfer receipt 2. Manager reviews the transfer details 3. Manager approves or declines the transfer 4. System updates payment status

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
BT01	Valid receipt image Amount= 5000 Valid bank details	Transfer marked as ‘pending’, manager notification sent	Refund completed successfully	Pass
BT02	Invalid receipt image Amount= 5000 Valid bank details	Error: Please upload valid image file	Partial refund completed	Fail
BT03	Valid receipt image Amount mismatch Uploaded= 5000 Expected = 6000 Valid bank details	Error: Managers declined the payment	Payment gets declined, Status ’failed’	Fail
BT04	Valid receipt image Amount= 5000 Valid bank details Managers approve	Error: Payment status updated ’completed’	Payment gets approve, Status ’completed’	Pass
BT05	Valid receipt image Amount= 5000 Valid bank details Manager declined	Error: Payment status updated ’failed’	Payment gets declined, Status ’failed’	Fail

Test Case Designed by: Kavisika K B P
Test Case Designed Date: October 4, 2025

Test Case ID: AI001
Testing Function: Automatic invoice generation after payment completion
Test Priority: High
Module Name: Invoice Management
Dependencies: Completed payment record
Test Steps: 1. Complete a payment transaction 2. Verify automatic invoice creation 3. Check invoice details accuracy 4. Verify email notification

Test ID	Test Input	Expected Output	Actual Output	Result(pass/fail)
AI01	Completed payment Amount: 5000 Type: membership User Id: valid	Invoice created with matching amount and user details	Invoice generated correctly	Pass
AI02	Completed payment Amount: 2500 Type: booking User Id: valid	Invoice created with matching amount and user details	Invoice shows booking details	Pass
AI03	Failed payment Type: membership User Id: valid	No invoice generated	No invoice created for failed payment	Fail
AI04	Completed payment Missing user data	Invoice creation fails gracefully	No invoice created for missing user data payment	Fail

Test Case Designed by: B DHAYABARI
Test Case Designed Date: October 4, 2025

Test Case ID:	TC-BKG-001
Testing Function:	CRUD operations for Bookings
Test Priority:	High
Module Name:	Booking management
Dependencies:	Database (Booking, Class, Trainer, Facility tables)

Test steps:

1.Add Booking → verify record inserted.
2.Add Booking with 0 seats → verify validation error.
3.Add Booking with past date → verify validation error.
4.Update Booking date/time/trainer/facility → verify record updated.
5.Cancel Booking → verify booking removed or marked cancelled.
6.Search/Filter Booking → verify filtered results displayed.
7.Generate Booking report → verify report contains correct data.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-BKG-001	Add booking (Yoga, John Doe, 05-10-2025,5.00pm, Fee 500)	Booking record inserted into DB with status = "Active"	Booking created successfully	Pass
TC-BKG-002	Add booking (0 seats)	Validation error: "No seats available"	System displayed validation error	Pass
TC-BKG-003	Add booking (past date)	Validation error: "Invalid booking date"	System displayed validation error	Pass
TC-BKG-004	Update booking date/time/facility	Booking record updated in DB	Booking updated	Pass

			successfully and reflected in calendar	
TC-BKG-005	Cancel booking	Booking removed or marked cancelled	Booking cancelled successfully	Pass
TC-BKG-006	Filter bookings by trainer/class/facility	Filtered list displayed	Correct filtered bookings displayed	Pass
TC-BKG-007	Generate daily/weekly/monthly report	Report shows correct booking data	Report generated correctly	Pass

Test Case ID:	TC-CLS-001
Testing Function:	CRUD operations for Classes
Test Priority:	High
Module Name:	Class management
Dependencies:	Database (Class table)

Test Steps:

8.Add Class → verify record inserted.
9.Add Class with missing required field → verify validation error.
10. Update Class details → verify record updated in DB.
11. Delete Class → verify record removed.
12. Search/Filter Class → verify filtered results displayed.

Test Case ID	**Input**	**Expected Output**	**Actual Output**	**Status**
TC-CLS-001	Add class (Yoga, 60 min, Fee 5000)	Class record inserted into DB with status = "Active"	Class record created successfully and visible in list	Fail
TC-CLS-002	Add class with empty Name	Validation error: "Class	System displayed validation error	Pass

		Name is required"		
TC-CLS-003	Update class duration/fee	Class record updated in DB	Updated successfully and reflected in list	Pass
TC-CLS-004	Delete class	Class removed from DB	Class removed from list	Pass
TC-CLS-005	Search class by Name	Filtered list displayed	Correct filtered classes displayed	Fail

Test Case ID:	TC-TRN-001
Testing Function:	CRUD operations for Trainers
Test Priority:	High
Module Name:	Trainer Management
Dependencies:	Database (Trainer table)

Test Steps:

1.Add Trainer → verify record inserted.
2.Add Trainer with missing required field → verify validation error.
3.Update Trainer specialization/status → verify record updated.
4.Delete Trainer → verify record removed.
5.Search/Filter Trainer → verify filtered results displayed.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-TRN-001	Add Trainer (John Doe, Yoga,1, Active)	Trainer record inserted in DB	Trainer created successfully	Pass

		with status = "Active"		
TC-TRN-002	Add Trainer without Name	Validation error: "Trainer Name is required"	System displayed validation error	Pass
TC-TRN-003	Update Trainer specialization/status	Trainer record updated in DB	Updated successfully and reflected in list	Fail
TC-TRN-004	Delete Trainer	Trainer removed from DB	Trainer removed from list	Pass
TC-TRN-005	Search Trainer by specialization	Filtered list displayed	Correct filtered trainers displayed	Fail

Test Case ID:	TC-FCL-001
Testing Function:	CRUD operations for Facilities
Test Priority:	High
Module Name:	Facility Management
Dependencies:	Database (Facility table)

Test Steps:

1.Add Facility → verify record inserted.
2.Add Facility with missing required field → verify validation error.
3.Update Facility details → verify record updated.
4.Delete Facility → verify record removed.
5.Search/Filter Facility → verify filtered results displayed.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-FCL-001	Add Facility (Room A, Equipment: Mats, Status Active)	Facility record inserted in DB	Facility created successfully	Pass

		with status = "Active"		
TC-FCL-002	Add Facility without Name	Validation error: "Facility Name is required"	System displayed validation error	Pass
TC-FCL-003	Update Facility equipment/status	Facility record updated in DB	Updated successfully and reflected in list	Fail
TC-FCL-004	Delete Facility	Facility removed from DB	Facility removed from list	Pass
TC-FCL-005	Search Facility by equipment	Filtered list displayed	Correct filtered facilities displayed	Fail

Test case Designed by: O.A.I. De Silva

Test case Designed Date: October 2, 2025

Test Case ID:	TC-TRK-001
Testing Function:	CRUD operations for progress
Test Priority:	High
Module Name:	Fitness Tracking – Nutrition Management
Dependencies:	Database (Nutrition, User tables)

Test steps:

1. Open Add Nutrition form, enter valid details → verify record inserted into DB.
2. Add Nutrition with missing required field (Name empty) → verify validation error shown.
3. Update nutrition calories value → verify DB record updated.
4. Delete Nutrition by ID → verify record removed.
5. Search Nutrition by food type → verify accurate results displayed.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-TRK-001	Add Nutrition (valid details: Chicken Breast, Protein: 31g, Carbs: 0g, Fat: 3.6g, Calories: 165)	Nutrition record inserted with status = "Active"	Record successfully created and visible in nutrition list	pass
TC-TRK-002	Add Nutrition (missing name)	Validation error "Food Name is required"	Validation message displayed	Pass
TC-TRK-003	Update Nutrition Calories	Record updated in DB	Calories updated successfully	Pass
TC-TRK-004	Delete Nutrition by ID	Record removed from DB	Record still visible due to soft delete issue	Pass
TC-TRK-005	Search by type "Protein Source"	Filtered nutrition list displayed	Filter returned accurate list	Fail

Test Case ID:	TC-TRK-006
Testing Function:	CRUD operations for Workout Management
Test Priority:	High
Module Name:	Fitness Tracking – Workout Management
Dependencies:	Database (Workout, User tables)

Test steps:

1. Open Add Workout form, enter valid details → verify workout record created.
2. Add Workout with missing duration → verify validation error.
3. Update workout intensity → verify record updated in DB.
4. Delete workout by ID → verify record removed.
5. Generate Workout summary report → verify correct data displayed.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-TRK-006	Add Workout (Push Ups, Duration: 30 mins, Intensity: Medium, CaloriesBurned: 200)	Workout record inserted	Workout successfully added	Pass
TC-TRK-007	Add Workout (missing Duration)	Validation error “Duration is required”	Validation message displayed	Pass
TC-TRK-008	Update Workout Intensity	Intensity updated successfully	Value updated in DB	Pass
TC-TRK-009	Delete Workout by ID	Record deleted from DB	Deleted successfully	Pass
TC-TRK-010	Generate Workout Report	Report generated with all recent workouts	Report generated with all recent workouts	Pass

Test Case ID:	TC- TRK-011
Testing Function:	Fitness Tracking – Goal Management
Test Priority:	High
Module Name:	Fitness Tracking – Goal Management
Dependencies:	Database (Goal, Progress tables)

Test steps:

6. Open Add Goal form, enter valid goal → verify record inserted.
7. Add Goal with missing target value → verify validation error.
8. Update goal status → verify DB record updated.
9. Delete goal by ID → verify record removed.

10. Search goal by type → verify correct results.

Test Case ID	Input	Expected Output	Actual Output	Status
TC-TRK-011	Add Goal (Type: Weight Loss, Target: 60kg, StartDate: 2025-02-01, EndDate: 2025-05-01)	Goal record created successfully	Record created and visible in goal list	Pass
TC-TRK-012	Add Goal (missing TargetValue)	Validation error "Target value is required"	Validation shown properly	Pass
TC-TRK-013	Update Goal Status → Completed	Status updated successfully in DB	Reflected correctly in progress dashboard	Pass
TC-TRK-014	Delete Goal by ID	Record removed	Record deleted successfully	Pass
TC-TRK-015	Search Goal by Type "Muscle Gain"	Filtered goal list displayed	Correct filtered data shown	Pass

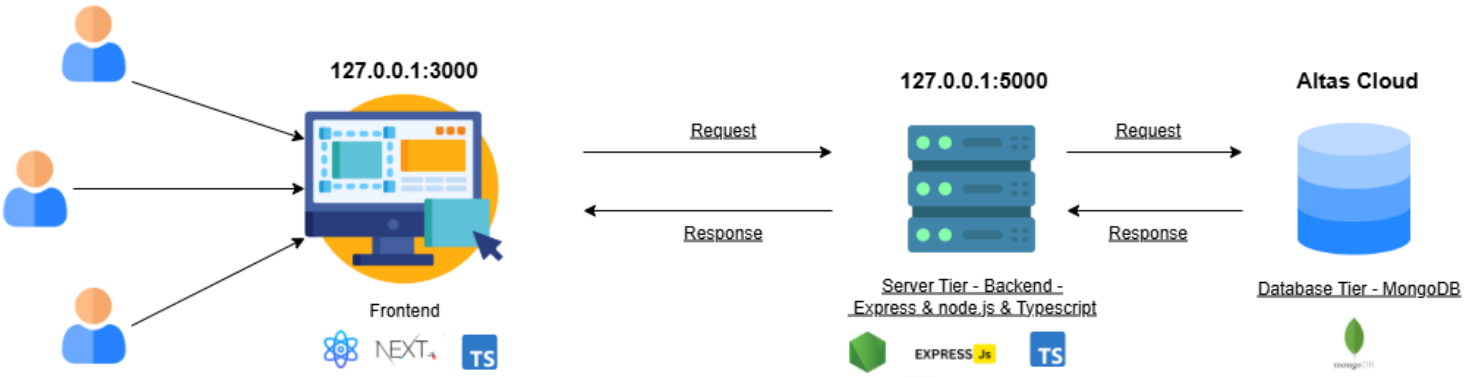
Test Case ID:	TC-TRK-016
Testing Function:	CRUD operations for Progress Tracking
Test Priority:	High
Module Name:	Fitness Tracking – Progress & Achievement Tracking
Dependencies:	Database (Progress, Goal, Workout, Nutrition tables)

Test steps:

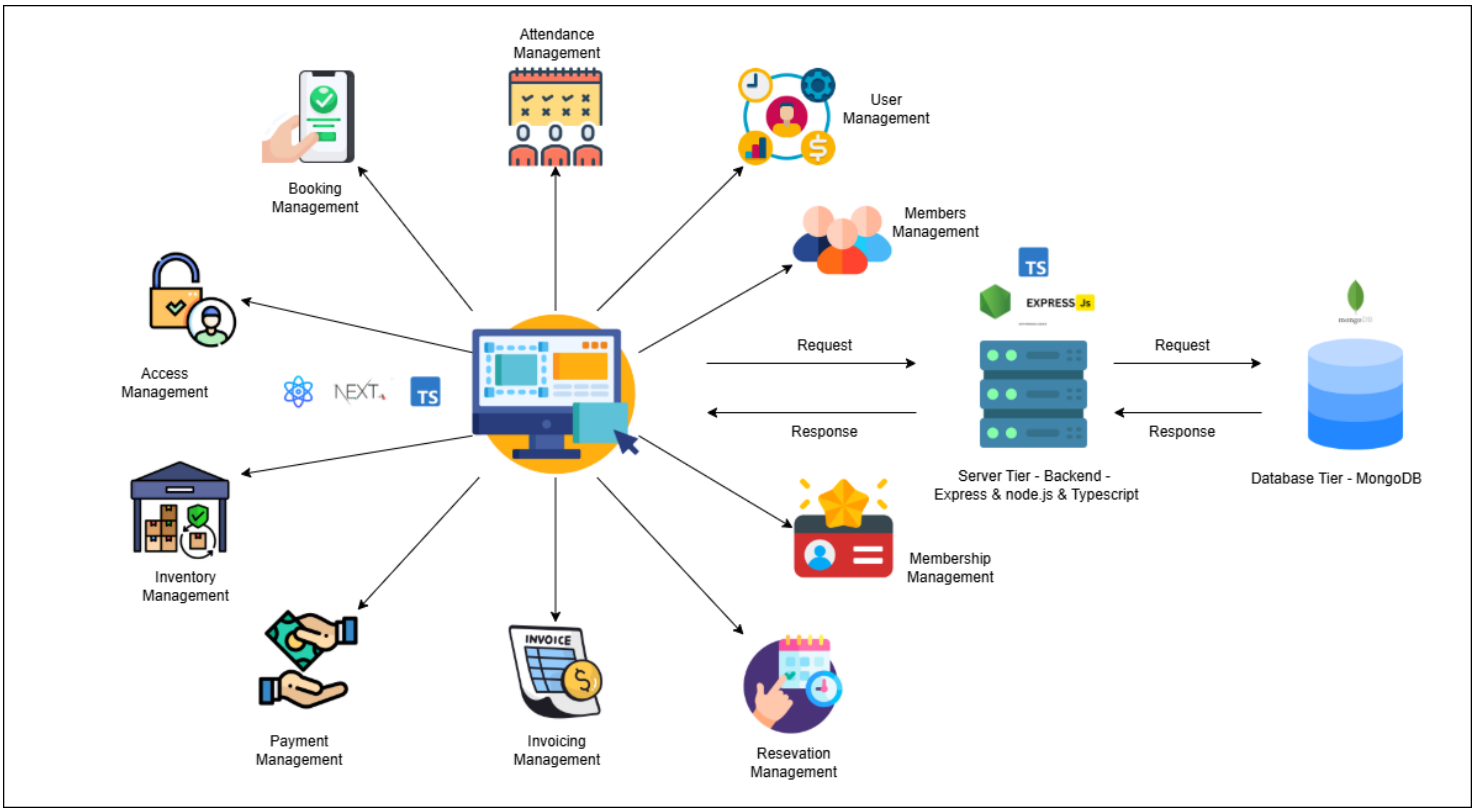
6. Add progress entry with linked goal and workout → verify record inserted.
7. Add Progress with missing date → verify validation error.
8. Update achievement rate → verify DB record updated.
9. View Progress summary → verify combined data from all linked entities.
10. Generate progress report → verify data accuracy.

Test Case ID	**Input**	**Expected Output**	**Actual Output**	**Status**
TC-TRK-016	Add Progress (Goal: Weight Loss, Date: 2025-02-05, AchievementRate: 80%)	Progress record linked to goal successfully	Progress saved successfully	Pass
TC-TRK-017	Add Progress (missing Date)	Validation error “Date is required”	Error displayed	Pass
TC-TRK-018	Update Achievement Rate	Record updated in DB	Updated correctly	Pass
TC-TRK-019	View Progress Summary	Dashboard shows combined goal, nutrition & workout info	Displayed correctly	Pass
TC-TRK-020	Generate Progress Report	Accurate report generated	Report generated correctly	Pass

5.Network Design



6.High-level system design diagram



7.Innovative parts of the project

ZFit is a high-end enterprise gym management system created to bring innovation and intelligent technology to streamline and automate fitness centers operations. The system brings a number of new features which make it unique compared to the traditional management solutions. It is also one of the pioneers of the QR Code Attendance and Access Control System wherein its members can check in without any hassle by using a customized QR code generated in their profile. This does not require any manual input or physical ID cards, which adds more security and gives real-time monitoring of gym attendance and facility utilization. The QR-based access system also ensures contactless and efficient members experience by supporting automatic door unlocking and real-time recording attendance.

The Calendar View Booking System is another significant innovation that allows users to book sessions, classes, and personal trainings using an intuitive interface. The availability is shown in real-time, and members are able to book slots right on the dashboard, as well as get instant confirmations or reminders. This intelligent scheduling reduces conflicts, increases the use of trainers and increases satisfaction of members due to transparency and convenience.

Combined, these elements make ZFit one of the smartest, technology-oriented solutions that connect the world of fitness management to automation and user-focused design. ZFit is redefining the way gyms operate and connect to their members in an interconnected, modern world by incorporating digital access control, real-time analytics and seamless booking.

8.Commercialization

ZFit gym management system is built to automate and computerize the running of a gym with key functionalities of the management system of a gym including membership management, reservation management, trainer management, facility management and fund management. The system is designed to be commercialized as a Software-as-a-Service (SaaS) platform which can be implemented by gyms, fitness centers, and wellness institutions of all sizes.

- The purpose of the introduction is to involve ZFit as a low-price, multi-service fitness center operation software.
- To automate the administration and financial functions, including member registration, bookings, payments, reporting.
- To bring about a centralized digital platform where members, trainers, managers, and owners can monitor activities and performance real-time.
- To make money by selling software, subscription packages and integrating services (such as PayHere online payments).

Members, staff, managers, trainers, accountants, health consultants, IT support staff, suppliers and owners are among the various stakeholders served by the ZFit system. It improves the efficiency of operations by automating them and supplying decision-makers with precise and real-time information. Basic features like online member registration, booking and rescheduling, QR based attendance, progress tracking, and secure payment processing add to enhanced user convenience and administrative accuracy.

The business model will mostly use a monthly or yearly subscription fee, which will be complemented by the development of unique software and third-party integrations (e.g., PayHere to handle payments). Further revenues can be achieved by partnering with fitness equipment manufacturers and nutritional brands, and by offering data-based information on business optimization.

Through the commercialization of the ZFit Gym Management System, a fitness center might reduce significantly its administration workload, decrease human error, and enhance customer satisfaction by utilizing digital creation. The system offers a scalable and cost-effective sustainable solution that has the potential to keep up with the increasing trend of digital transformation in the fitness industry.

9.Individual Contribution

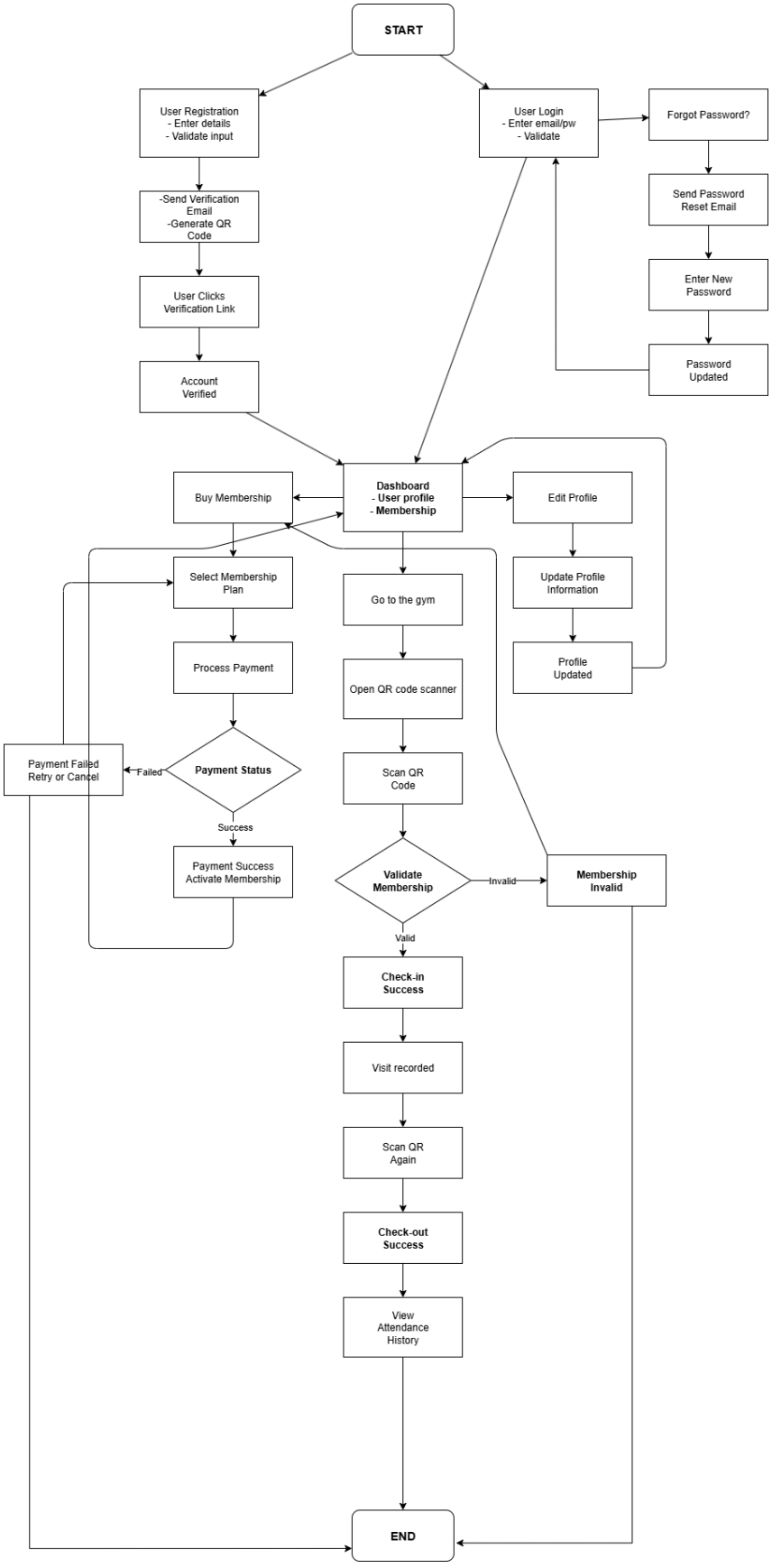
	IT Number	Student Name	Contribution
1	IT 23 5621 10	H R G N Karunathilaka	Er diagram and test cases
2	IT 23 5418 70	N M B M L B Nawaratne	Normalized schema and test cases
3	IT 23 5804 80	K B P Kavisika	Network diagram and test cases
4	IT 23 7536 62	O A l De Silva	Commercialization and test cases
5	IT 23 7414 78	B Dhayabari	Commercialization and test cases

a.User, Attendance and Membership Management – IT23562110 – Karunathilaka H R G N

IT23562110 – Karunathilaka H R G N	Completion Level of Features	
Completed	User registration and login	100%
	Email verification	100%
	Password reset with email reset link	100%
	Search and filter users	100%
	Generate unique qr code automatically for each user	100%
	Edit user profiles	100%
	Scan qr code to track user attendance and access	100%
	Generate user reports	100%
	Generate attendance reports	100%
	Generate membership reports	100%

	CRUD operations in membership plan	100%
	Add new users with force password reset onboarding flow	100%
To be Completed	Force check in for members with no memberships	70%
	Renew memberships for members	60%
	Restrict access based on user roles	70%
	Frontend & Backend Validations	90%

Flow Chart


START
User Registration
- Enter details
- Validate input
User Login
- Enter email/pw
- Validate
Forgot Password?
-Send Verification Email
-Generate QR Code
Send Password Reset Email
User Clicks Verification Link
Enter New Password
Account Verified
Password Updated
Dashboard
- User profile
- Membership
Buy Membership
Edit Profile
Select Membership Plan
Go to the gym
Update Profile Information
Process Payment
Open QR code scanner
Profile Updated
Payment Failed
Retry or Cancel
Failed
Payment Status
Scan QR Code
Success
Payment Success
Activate Membership
Validate Membership
Invalid
Membership Invalid
Valid
Check-in Success
Visit recorded
Scan QR Again
Check-out Success
View Attendance History
END

ZFit System Pseudocode

1. Authentication Flow

Registration & Email Verification

```
FUNCTION registerUser(email, password, name):
   VALIDATE email format and password length
   CHECK if email already exists → RETURN error
   HASH password
   SAVE user {email, hashedPassword, name, emailVerified: false}
   token = GENERATE verification token
   SEND email with verification link + token
   RETURN success
END

FUNCTION verifyEmail(token):
   VALIDATE token → GET userId
   UPDATE user SET emailVerified = true
   RETURN success
END
```

Login & Password Reset

```
FUNCTION login(email, password):
   user = FIND by email
   VERIFY password hash
   CHECK emailVerified = true
   sessionToken = GENERATE session token
   SAVE session {userId, token, expiresAt}
   RETURN {token, userData}
END

FUNCTION forgotPassword(email):
   user = FIND by email
   resetToken = GENERATE token with 1hr expiry
   SEND reset email with token
END

FUNCTION resetPassword(token, newPassword):
   VALIDATE token → GET userId
```

```
    hashedPassword = HASH newPassword
    UPDATE user SET password = hashedPassword
    DELETE all user sessions
    RETURN success
END
```

2. Profile & Membership Management

Profile Operations

```
FUNCTION getDashboard(sessionToken):
    userId = VALIDATE session
    userData = GET user by userId
    membership = GET active membership
    attendanceHistory = GET recent 10 records
    RETURN {userData, membership, attendanceHistory}
END

FUNCTION editProfile(sessionToken, updates):
    userId = VALIDATE session
    VALIDATE updates (name, phone, etc.)
    UPDATE user SET updates
    RETURN updated user data
END
```

Membership Purchase

```
FUNCTION buyMembership(sessionToken, planId, paymentDetails):
    userId = VALIDATE session
    plan = GET plan by planId
    CHECK no active membership exists

    payment = CREATE {userId, planId, amount, status: "pending"}
    result = CALL paymentGateway.charge(paymentDetails, plan.price)

    IF result.success THEN
        UPDATE payment SET status = "success"
        CREATE membership {
            userId, planId,
            startDate: today,
            endDate: today + plan.duration,
            status: "active"
        }
        RETURN success
    ELSE
```

```
      UPDATE payment SET status = "failed"
      RETURN error "Payment failed"
   END IF
END
```

3. Attendance System

QR Code Generation

```
FUNCTION generateQR(sessionToken):
   userId = VALIDATE session
   membership = GET active membership
   CHECK membership exists and not expired

   sessionId = GENERATE unique ID
   qrData = {sessionId, userId, timestamp, expiresAt: now + 5min}
   qrCode = GENERATE QR image from qrData
   SAVE qrData to cache

   RETURN qrCode
END
```

Check-In & Check-Out

```
FUNCTION checkIn(qrData):
   PARSE qrData → GET {sessionId, userId}
   VALIDATE QR session exists and not expired
   membership = GET active membership for userId
   CHECK membership valid and not expired
   CHECK no active attendance session

   CREATE attendance {
      userId,
      checkInTime: now,
      checkOutTime: null,
      status: "active"
   }
   RETURN success "Checked in successfully"
END

FUNCTION checkOut(qrData):
   PARSE qrData → GET userId
   attendance = GET active attendance session
   CHECK attendance exists
```

```
  UPDATE attendance SET {
    checkOutTime: now,
    duration: now - checkInTime,
    status: "completed"
  }
  RETURN success "Checked out successfully"
END
```

Attendance History

```
FUNCTION getAttendanceHistory(sessionToken, page, limit):
  userId = VALIDATE session
  offset = (page - 1) * limit

  records = GET attendance WHERE userId
        ORDER BY checkInTime DESC
        LIMIT limit OFFSET offset

  FOR each record DO
    record.duration = checkOutTime - checkInTime
  END FOR

  RETURN records
END
```

4. Helper Functions

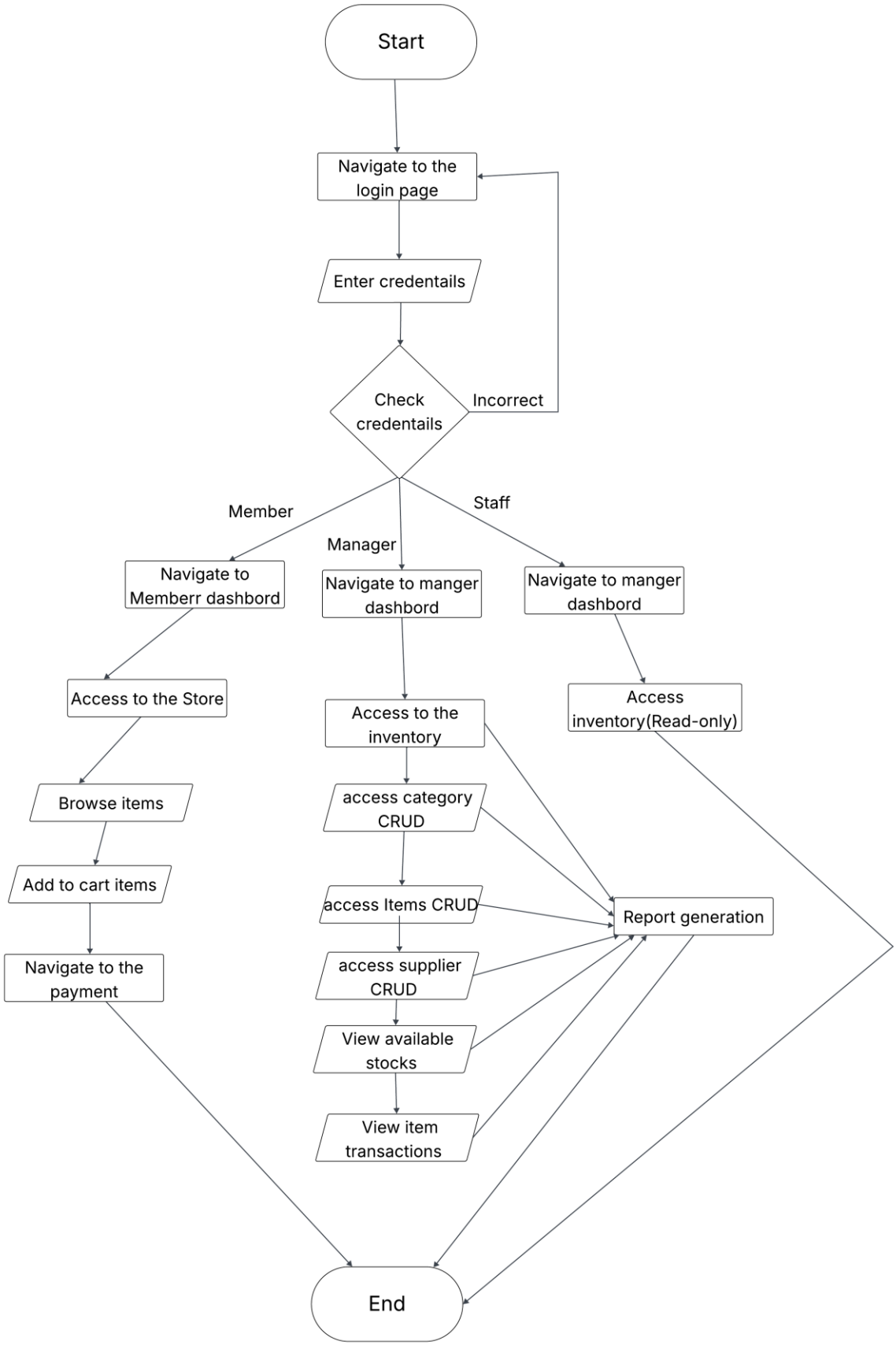
```
FUNCTION validateSession(token):
  session = FIND by token
  CHECK session.expiresAt > now
  RETURN session.userId
END

FUNCTION hashPassword(password):
  RETURN bcrypt.hash(password, saltRounds: 10)
END

FUNCTION generateToken(data, expiresIn):
  RETURN JWT.sign(data, secretKey, {expiresIn})
END
```

b.Inventory Control – IT 23 5418 70 – Nawaratne N M B M L B

IT23698536 Nawaratne N M B M L B	Completion Level of Features	
Completed	CRUD operations in categories	100%
	CRUD operations in items	100%
	CRUD operations in suppliers	100%
	Search and filter inventory (Category, Items have sellable and equipment tabs, supplier, stock)	100%
	View Stock details for sellable items	100%
	Show low stock alerts	100%
	Generate inventory reports	100%
	Create store and the cart	100%
	Automatically update stock levels in real time whenever changes occur	100%
	Gym equipment maintenance tracking.	100%
	Members can browse available gym products and add them to their cart	100%

To be Completed	Managers are notified via email when stock levels are low.	60%
	Track all inventory transactions.	70%
	Allow members to view their complete order history.	60%
	Restrict inventory access based on user roles	70%
	Frontend & Backend Validations	90%

Flow Chart


Start
Navigate to the login page
Enter credentails
Check credentails
Incorrect
Member
Manager
Staff
Navigate to Memberr dashbord
Navigate to manger dashbord
Navigate to manger dashbord
Access to the Store
Access to the inventory
Access inventory(Read-only)
Browse items
access category CRUD
Add to cart items
access Items CRUD
Report generation
Navigate to the payment
access supplier CRUD
View available stocks
View item transactions
End

Pseudocode

```
PROGRAM GymInventorySystem
FUNCTION main()
   DISPLAY "Navigate to login page"
   REPEAT
      credentials ← INPUT_CREDENTIALS()
      IF NOT isValid(credentials) THEN
         DISPLAY "Incorrect credentials"
      END IF
   UNTIL isValid(credentials)
   role ← getUserRole(credentials.username)      // "Manager" | "Staff" | "Member"
   IF role = "Manager" THEN
      managerFlow()
   ELSE IF role = "Staff" THEN
      staffFlow()
   ELSE IF role = "Member" THEN
      memberFlow()
   ELSE
      DISPLAY "Unknown role"
   END IF
END FUNCTION

// MANAGER //
FUNCTION managerFlow()
   DISPLAY "Navigate to Manager dashboard"
   WAIT_FOR "Click: Inventory"
   managerInventoryMenu()
   DISPLAY "End"
END FUNCTION

FUNCTION managerInventoryMenu()
   DO
      DISPLAY "
        Manager Inventory:
        1) Categories (CRUD)
        2) Items (Saleable & Equipment) (CRUD)
        3) Suppliers (CRUD)
        4) View Stock Table
        5) View Inventory Transactions
        6) Generate Reports
```

```
        7) Logout / Back
      "
choice ← INPUT()
      SWITCH choice
        CASE 1:
          categoryCRUD()
          generateReportsOption()       // per flow: reports available from each module
        CASE 2:
          itemsCRUD()                   // includes low-stock alert check on updates
          generateReportsOption()
        CASE 3:
          suppliersCRUD()
          generateReportsOption()
        CASE 4:
          viewStockTable()
          generateReportsOption()
        CASE 5:
          viewInventoryTransactions()
          generateReportsOption()
        CASE 6:
          generateReports("Manager")
        CASE 7:
          EXIT_DO
        DEFAULT:
          DISPLAY "Invalid choice"
      END SWITCH
    LOOP
END FUNCTION
FUNCTION categoryCRUD()
    // Create / Read / Update / Delete categories
    // Validate inputs; audit all changes
END FUNCTION
FUNCTION itemsCRUD()
    // Create / Read / Update / Delete items
    // Item has: type ∈ {Saleable, Equipment}, common fields, plus type-specific fields
    // On any quantity change or receipt/sale adjustment:
    FOR EACH changedItem IN pendingItemChanges
      APPLY_CHANGE(changedItem)
      checkLowStockAndAlert(changedItem)  // <-- low-stock hook
    END FOR
```

```
END FUNCTION
FUNCTION suppliersCRUD()
   // Create / Read / Update / Delete suppliers
   // Block hard delete if referenced by active POs; otherwise allow archive
END FUNCTION
FUNCTION viewStockTable()
   // Read-only tabular view of item quantities, reorder levels, statuses
END FUNCTION
FUNCTION viewInventoryTransactions()
   // Read-only list of movements (sales, receipts, adjustments) with filters
END FUNCTION
FUNCTION generateReportsOption()
   IF CONFIRM("Generate report for this module?") THEN
      generateReports("Manager")
   END IF
END FUNCTION
FUNCTION generateReports(requestorRole)
   // Build CSV/PDF for Summary / Low Stock / Maintenance / Transactions / Suppliers
   // Apply selected filters; timestamp and audit the download
   DISPLAY "Report generated"
END FUNCTION
FUNCTION checkLowStockAndAlert(item)
   IF item.type = "Saleable" AND item.quantity ≤ item.reorderLevel THEN
      createLowStockAlert(item)          // email + UI badge
   END IF
END FUNCTION
FUNCTION createLowStockAlert(item)
   // Throttle duplicates (e.g., once per 24h per item unless recovered)
   SEND_EMAIL(to=Managers, subject="Low Stock: " + item.name, body=...)
   SET_UI_BADGE(item, "LOW STOCK")
END FUNCTION

// STAFF (READ-ONLY) //
FUNCTION staffFlow()
   DISPLAY "Navigate to Staff dashboard"
   WAIT_FOR "Open Inventory (Read-only)"
   staffInventoryMenu()
   DISPLAY "End"
END FUNCTION
FUNCTION staffInventoryMenu()
```

```
    DO
        DISPLAY "
          Staff Inventory (Read-only):
          1) View Categories
          2) View Items (Saleable & Equipment)
          3) View Suppliers
          4) View Stock Table
          5) View Inventory Transactions
          6) Generate Reports
          7) Logout / Back
"
        choice ← INPUT()
        SWITCH choice
            CASE 1: viewCategories()
            CASE 2: viewItems()
            CASE 3: viewSuppliers()
            CASE 4: viewStockTable()
            CASE 5: viewInventoryTransactions()
            CASE 6: generateReports("Staff")
            CASE 7: EXIT_DO
            DEFAULT: DISPLAY "Invalid choice"
        END SWITCH
    LOOP
END FUNCTION
FUNCTION viewCategories(); END FUNCTION
FUNCTION viewItems(); END FUNCTION
FUNCTION viewSuppliers(); END FUNCTION

/// MEMBER (STORE) //
FUNCTION memberFlow()
    DISPLAY "Navigate to Member dashboard"
    WAIT_FOR "Access the Store"
    // Simple linear flow per chart
    browseItems()
    addItemsToCart()
    navigateToPayment()
    DISPLAY "End"
END FUNCTION
FUNCTION browseItems()
    // List only Saleable items with price and availability
```

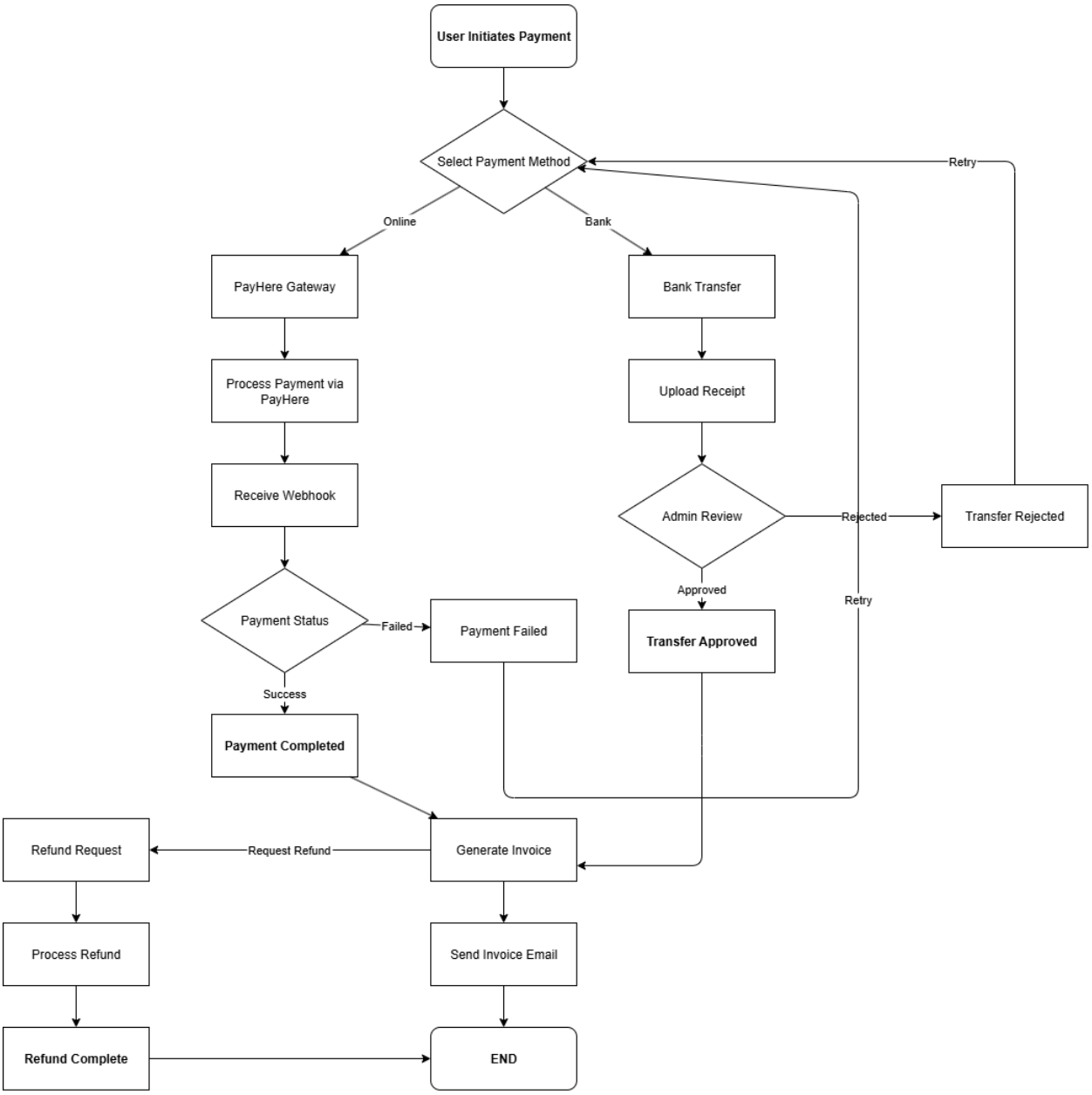
```
END FUNCTION
FUNCTION addItemsToCart()
   // Add selected Saleable items; enforce stock limits
END FUNCTION
FUNCTION navigateToPayment()
   // Proceed to checkout and payment
END FUNCTION
```

c.Payment, Invoice, Refund, and Bank Transfer Management – IT23580480 – Kavisika K B P

IT 23580480- Kavisika K B P	Completion Level of Features	
Completed	Payment processing via PayHere gateway	100%
	Webhook handling for payment status updates	100%
	Invoice automatic generation after payment completion	100%
	Refund processing for completed payments	100%
	Bank transfer receipt upload and validation	100%
	Payment history and filtering	100%
	Generate Payment reports	100%
	Generate Refund reports	100%
	Generate invoice reports	100%
	Bank transfer admin approval workflow	100%

	Payment reports with PDF export	100%
	Frontend and backend integrations	100%
To be Completed	Role-based access restrictions for payments	80%
	Membership renewal payment automation	50%
	Force payment retry for failed transactions	70%
	Frontend Validations and error handling	80%

Flow chart


User Initiates Payment
Select Payment Method
Retry
Online
Bank
PayHere Gateway
Bank Transfer
Process Payment via PayHere
Upload Receipt
Receive Webhook
Admin Review
Rejected
Transfer Rejected
Approved
Retry
Payment Status
Failed
Payment Failed
Transfer Approved
Success
Payment Completed
Refund Request
Request Refund
Generate Invoice
Process Refund
Send Invoice Email
Refund Complete
END

Payment System Pseudocode

1. Create Payment

```
FUNCTION createPayment(amount, type, method, userId, relatedId):
   VALIDATE amount > 0 and required fields
   transactionId = GENERATE unique ID
   payment = CREATE {transactionId, amount, type, method, userId, relatedId, status:
"pending"}
   SAVE payment to database
   RETURN payment
END
```

2. PayHere Gateway

Process Payment

```
FUNCTION processPayHerePayment(paymentData, customerInfo):
   VALIDATE customer details
   request = PREPARE PayHere request {
      merchant_id, order_id, amount, currency,
      customer_name, email, phone
   }
   response = SEND to PayHere API
   RETURN response.checkout_url
END
```

Handle Webhook

```
FUNCTION handlePayHereWebhook(webhookData):
   VERIFY signature with merchant secret
   payment = FIND by order_id

   IF webhookData.status == "2" THEN
      UPDATE payment SET status = "completed"
   ELSE
      UPDATE payment SET status = "failed", failureReason = webhookData.message
   END IF

   SAVE webhookData to payment.gatewayResponse
   RETURN success
END
```

3. Bank Transfer

Create Transfer

```
FUNCTION createBankTransfer(userId, membershipId, amount, receiptFile):
   CHECK no pending transfer exists for userId
   receiptUrl = UPLOAD receipt to storage
```

```
    transfer = CREATE {
        userId, membershipId, amount, receiptUrl,
        status: "pending"
    }
    SAVE transfer to database
    NOTIFY admin for approval
    RETURN transfer
END
```

Approve Transfer

```
FUNCTION approveBankTransfer(transferId, adminId, notes):
    transfer = FIND by transferId
    CHECK transfer.status == "pending"

    UPDATE transfer SET status = "approved", approvedBy = adminId, notes
    payment = CREATE payment record from transfer
    SEND approval email to user
    RETURN success
END
```

4. Invoice Generation

```
FUNCTION generateInvoice(paymentId, userId, items):
    payment = FIND by paymentId
    user = FIND by userId

    subtotal = SUM items.price
    tax = subtotal * 0.1
    total = subtotal + tax

    invoiceNumber = GENERATE "ZF-INV-" + YEAR + "-" + SEQUENCE

    invoice = CREATE {
        invoiceNumber, paymentId, userId,
        items, subtotal, tax, total,
        status: "sent"
    }
    SAVE invoice to database
    RETURN invoice
END
```

5. Refund Process

```
FUNCTION processRefund(paymentId, amount, reason):
```

```
    payment = FIND by paymentId
    CHECK payment.status == "completed"
    CHECK amount <= payment.amount - payment.refundedAmount

    UPDATE payment SET refundedAmount += amount

    IF payment.refundedAmount == payment.amount THEN
        UPDATE payment SET status = "refunded"
    END IF

    LOG refund {paymentId, amount, reason, timestamp}
    RETURN success
END
```

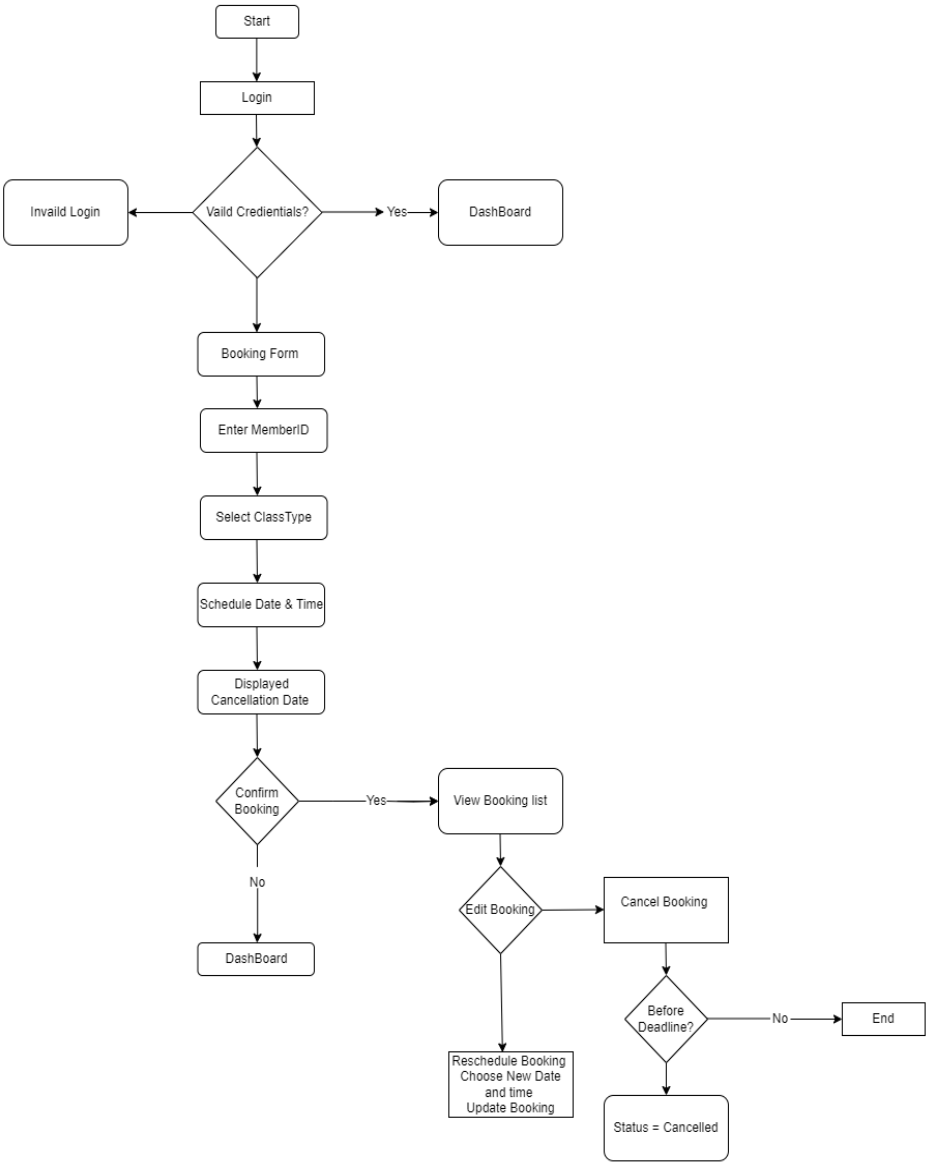
6. Get Payment Status

```
FUNCTION getPaymentStatus(transactionId):
    payment = FIND by transactionId
    RETURN {
        transactionId,
        status: payment.status,
        amount: payment.amount,
        method: payment.method,
        createdAt: payment.createdAt
    }
END
```

d.Bookinfg and Reservation Management

IT23741478 – B DHAYABARI	**Completion Level of Features**	
Completed	Booking Creation	100%
	View All Bookings	100%
	Crud Operations for Booking	100%
	Crud Operations for Classes	100%
	Crud Operations for Trainers	100%
	Crud Operations for Facilities	100%
	Validations for all Crud Operations (Zod + Mongoose)	100%
	Class Type Handling	100%
	Search and filter users	100%
	Calendar Access (Data-Level)	100%
	Booking Rescheduling	100%
To be Completed	Email / Notification Alerts	60%
	Generate Booking and Reservation Report	30%
	User-based access control	50%
	Frontend Validations	80%

Flow chart


Start
Login
Vaild Credientials?
Invaild Login
Yes
DashBoard
Booking Form
Enter MemberID
Select ClassType
Schedule Date & Time
Displayed Cancellation Date
Confirm Booking
Yes
View Booking list
No
DashBoard
Edit Booking
Cancel Booking
Reschedule Booking
Choose New Date and time
Update Booking
Before Deadline?
No
End
Status = Cancelled

Pseudocode

```
BEGIN
// --- Booking Management System ---
// --- CRUD for Classes ---
FUNCTION ManageClasses()
   DISPLAY all classes in table
   REPEAT
      INPUT action (Add / Edit / Delete / Exit)
      IF action == "Add" THEN
         INPUT className, classType, duration, seats, fee
         IF seats <= 0 OR seats > 20 THEN
            DISPLAY "Seats must be 1-20"
         ELSE IF duration < 40 THEN
            DISPLAY "Duration must be at least 40 minutes"
         ELSE IF fee <= 0 THEN
            DISPLAY "Fee cannot be zero or negative"
         ELSE
            SAVE class to database
            DISPLAY "Class added successfully"
         END IF
      ELSE IF action == "Edit" THEN
         INPUT classID
         FETCH class by ID
         INPUT updatedData
         APPLY same validation as Add
         UPDATE class in database
         DISPLAY "Class updated successfully"
      ELSE IF action == "Delete" THEN
         INPUT classID
         DELETE class from database
         DISPLAY "Class deleted successfully"
      END IF
   UNTIL action == "Exit"
END FUNCTION

// --- CRUD for Trainers ---
FUNCTION ManageTrainers()
   DISPLAY all trainers in table
   REPEAT
      INPUT action (Add / Edit / Delete / Exit)
```

```
    IF action == "Add" THEN
      INPUT trainerName, specialization, experience
      IF experience <= 0 THEN
        DISPLAY "Experience cannot be zero"
      ELSE
        SAVE trainer to database
        DISPLAY "Trainer added successfully"
      END IF
    ELSE IF action == "Edit" THEN
      INPUT trainerID
      FETCH trainer by ID
      INPUT updatedData
      APPLY same validation as Add
      UPDATE trainer in database
      DISPLAY "Trainer updated successfully"
    ELSE IF action == "Delete" THEN
      INPUT trainerID
      DELETE trainer from database
      DISPLAY "Trainer deleted successfully"
    END IF
  UNTIL action == "Exit"
END FUNCTION

// --- CRUD for Facilities ---
FUNCTION ManageFacilities()
  DISPLAY all facilities in table
  REPEAT
    INPUT action (Add / Edit / Delete / Exit)
    IF action == "Add" THEN
      INPUT facilityName, capacity, equipment
      IF equipment < 0 THEN
        DISPLAY "Equipment cannot be negative"
      ELSE
        SAVE facility to database
        DISPLAY "Facility added successfully"
      END IF
    ELSE IF action == "Edit" THEN
      INPUT facilityID
      FETCH facility by ID
      INPUT updatedData
```

```
            APPLY same validation as Add
            UPDATE facility in database
            DISPLAY "Facility updated successfully"
        ELSE IF action == "Delete" THEN
            INPUT facilityID
            DELETE facility from database
            DISPLAY "Facility deleted successfully"
        END IF
    UNTIL action == "Exit"
END FUNCTION

// --- Bookings (Member) ---
FUNCTION ManageBookings(userRole)
    DISPLAY booking table filtered by userRole
    REPEAT
        INPUT action (Book / Cancel / Reschedule / Exit)
        IF action == "Book" THEN
            INPUT className, trainerName, facilityName, bookingDate, bookingTime
            FETCH IDs for selected names
            IF bookingDate < currentDate THEN
                DISPLAY "Booking must be in the future"
            ELSE
                SAVE booking to database
                DISPLAY "Booking successful"
            END IF
        ELSE IF action == "Cancel" THEN
            INPUT bookingID
            FETCH booking by ID
            IF currentDate >= bookingDate THEN
                DISPLAY "Cannot cancel after booking date"
            ELSE
                UPDATE booking status to "Cancelled"
                DISPLAY "Booking cancelled"
            END IF
        ELSE IF action == "Reschedule" THEN
            INPUT bookingID, newDate, newTime
            IF newDate < currentDate THEN
                DISPLAY "Cannot reschedule to past date"
            ELSE
                UPDATE booking date/time in database
```

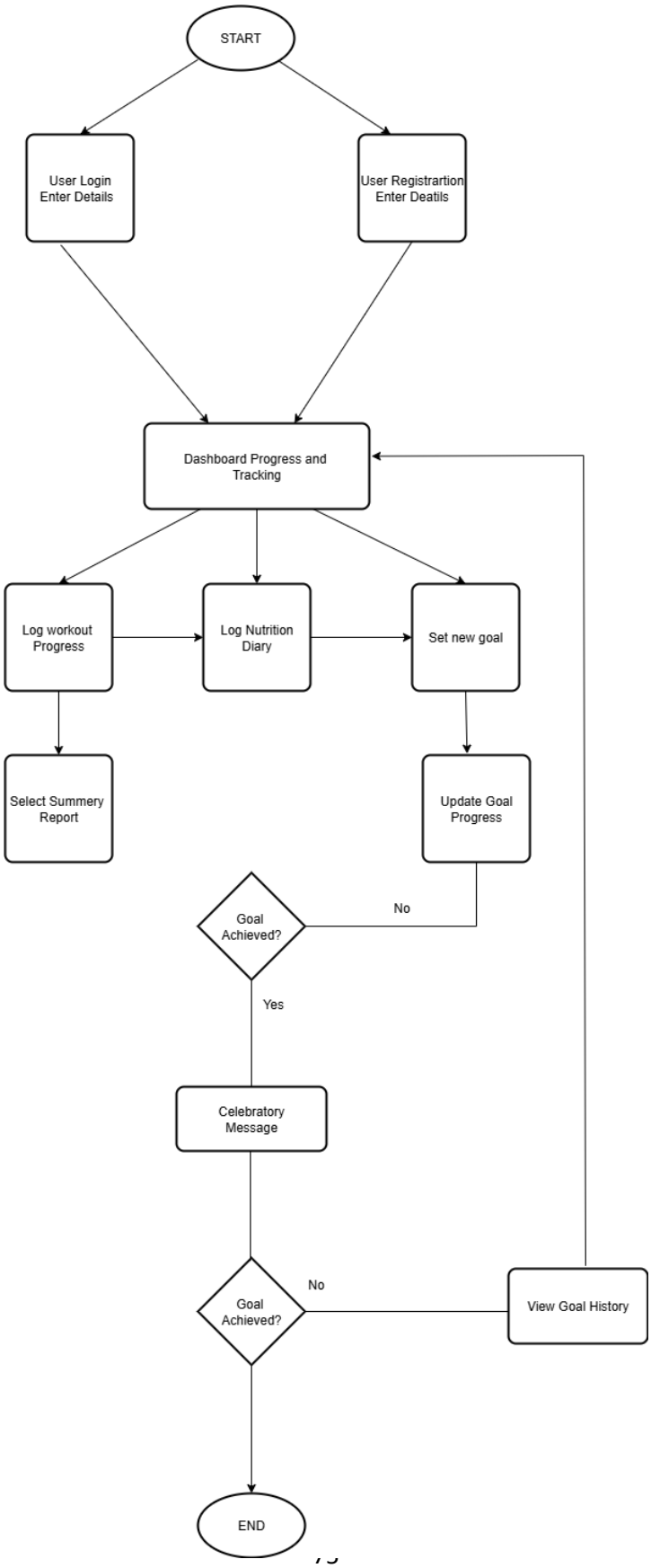
```
            DISPLAY "Booking rescheduled"
          END IF
        END IF
    UNTIL action == "Exit"
END FUNCTION

// --- Calendar View ---
FUNCTION ViewCalendar(userRole, selectedDate)
    IF userRole == "Member" THEN
        DISPLAY all bookings for current user on selectedDate
    ELSE IF userRole == "Admin" OR userRole == "Manager" THEN
        DISPLAY all bookings for all members on selectedDate
    END IF
END FUNCTION

// --- Reports (Admin/Manager) ---
FUNCTION GenerateReports(filterType)
    FETCH bookings from database based on filterType (daily/weekly/monthly/yearly)
    DISPLAY report table with:
        Booking ID | Member Name | Class | Trainer | Facility | Date | Time | Fee
END FUNCTION

END
```

e.Function Tracking and Progress Management -IT23753662 – O.A.I. De Silva

IT23753662 – O.A.I De Silva	Completion Level of Features	
Completed	CRUD operations for Workout Management (Add, View, Update, Delete)	100%
	CRUD operations for Nutrition Management (Add, View, Update, Delete)	100%
	CRUD operations for Progress Management	100%
	CRUD operations for Goal Management	100%
	Validation for all CRUD operations (Zod + Mongoose Schema)	100%
	Generate fitness progress reports (individual and overall)	100%
	Display member progress graphs (Workout + Nutrition trends)	100%
	Integration of user-based access control (Members only)	100%
	Integration with member dashboard to display progress data	100%
	Link Progress module with Goal, Workout, and Nutrition entities	100%

	Backend API routes for all four tracking modules	100%
	Tested all CRUD functionalities in Postman and MongoDB	100%
	Report generation (PDF/Graph) with summary analytics	100%
To be Completed	Email notification for weekly progress summary	70%
	Real-time dashboard charts and visual insights (Frontend)	60%
	Role-based restrictions for admin vs. member view access	70%
	UI/UX enhancements and responsive design for progress pages	85%

Flow Chart


START
User Login
Enter Details
User Registrartion
Enter Deatils
Dashboard Progress and
Tracking
Log workout
Progress
Log Nutrition
Diary
Set new goal
Select Summery
Report
Update Goal
Progress
Goal
Achieved?
No
Yes
Celebratory
Message
Goal
Achieved?
No
View Goal History
END

Pseudocode

```
// --- Login ---
BEGIN
   REPEAT
      INPUT username, password
      IF credentials are valid THEN
         DISPLAY "Login Successful"
         PROCEED to Progress Management Dashboard
      ELSE
         DISPLAY "Invalid username or password"
      END IF
   UNTIL credentials are valid

// --- Progress Management Dashboard ---
DISPLAY " Tracking and Progress"

REPEAT
   DISPLAY menu: [Add Progress] [Update Progress] [Delete Progress] [Generate Report]
[Logout]
   INPUT choice

   IF choice = "Add Progress" THEN
      INPUT date, workout_type, duration, calories_burned, notes
      VALIDATE inputs
      IF valid THEN
         STORE record in database
         DISPLAY "Progress Added Successfully"
      ELSE
         DISPLAY "Invalid Input, Please Try Again"
      END IF

   ELSE IF choice = "Update Progress" THEN
      SELECT record_to_update
      INPUT updated_values
      VALIDATE updated_values
      IF valid THEN
         UPDATE record in database
         DISPLAY "Progress Updated Successfully"
      END IF
```

```
    ELSE IF choice = "Delete Progress" THEN
        SELECT record_to_delete
        CONFIRM deletion
        IF confirmed THEN
            DELETE record from database
            DISPLAY "Progress Deleted Successfully"
        END IF

    ELSE IF choice = "Generate Report" THEN
        RETRIEVE all progress records
        CALCULATE summaries and averages
        EXPORT or DISPLAY report
    END IF
UNTIL choice = "Logout"
END FUNCTION
END
```