IT2080 – ITP

Year 2 Semester 2 2024

Assignment 2

Surfing School Management System

ITP24J_B04_07

Batch: 4.1

Table of Contents

01.Progress

Function	Completed User Stories	In completed stories
Lesson Management (IT22127778)	• As an instructor, I want to create new lessons so that I can offer different surf lessons to students. • As an instructor, I want to update existing lessons so that I can keep the information current. • As an instructor, I want to delete a lessons so that I can remove outdated or irrelevant lessons. • As a student, I want to view available surfing packages so that I can choose a suitable lesson to book. • As a student, I want to view my scheduled lessons so that I can check it in lessons section.	As an instructor, I want to log in to the system so that I can access and manage my lessons under packages.

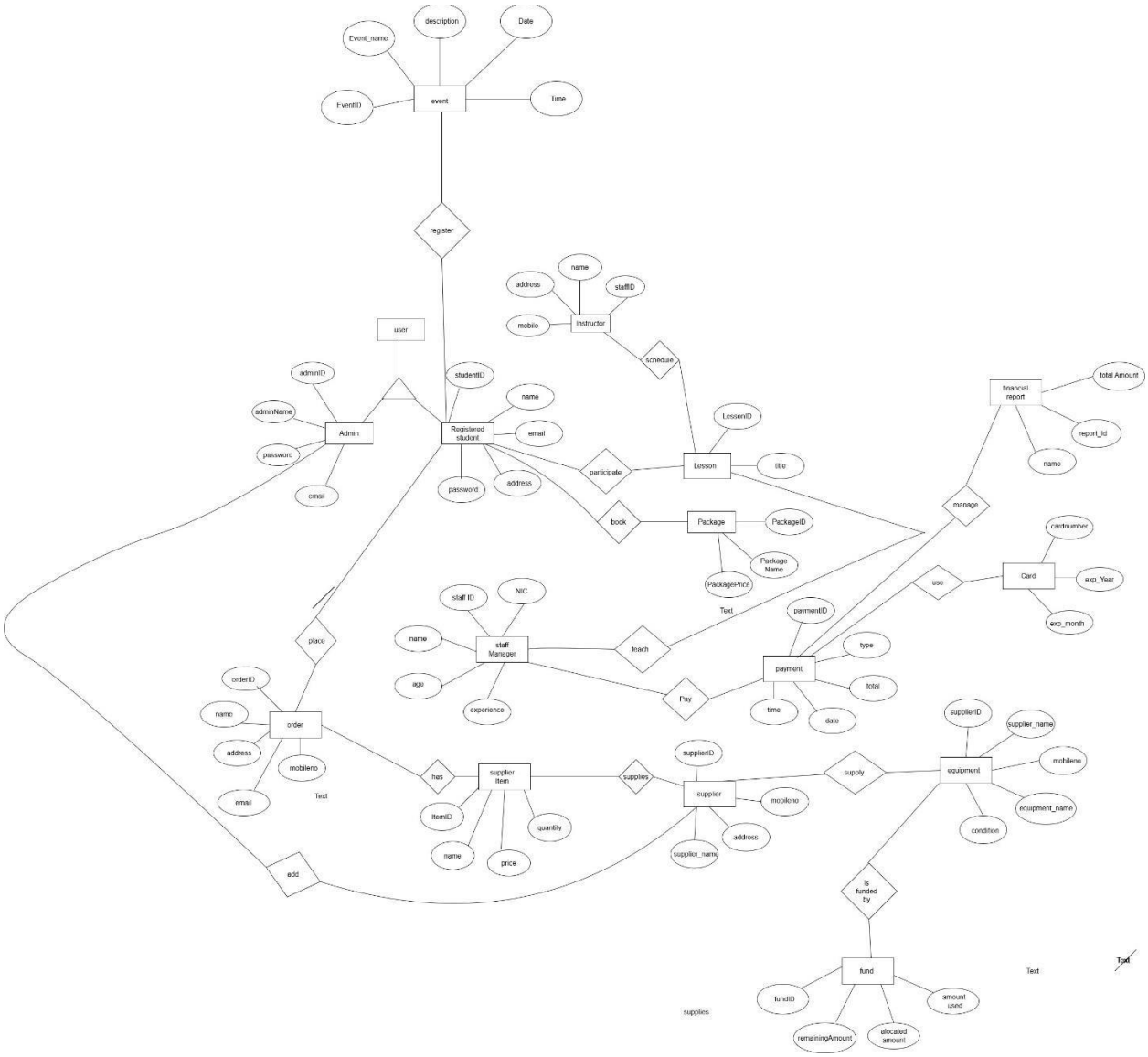
Event Planning (IT22243980)	• As a event manager, I need a user register to an event that they want, so then I can get registered details. • As a event manager, I need update the registered event details when the students need to change events. • As a event manager, I need to delete event details, so when the details were saved and event ended I want to delete them. • As a event manager, I want search student name and get the event that he registered. • As a event manager, I want to get a report as a pdf file about the daily registered	As a event manager, I want to download registration form for students from user side.
Payment (IT22165848)	• **As a customer**, I want to securely make payments for my surf lessons or equipment so that I can complete my booking without any hassle. • **As a customer,** I want to be able to save my personal details to the system according to the personal details and edit and delete my personal details. • **As a financial manager**, I want to generate reports on all payments so that I can analyze the financial performance of the surfing school • **As a financial manager**, I want to manage user payments, including updating, deleting, and viewing specific transaction details so that I can ensure the system is up to date with accurate data	**As a Customer,** I want to Pay the bill through PayPal Gateway.

Equipment maintenance (IT22083814)	• As an Equipment Manager, I want to add new equipment to the system with its full details so that I can track it accurately. • As an Equipment Manager, I want to access detailed information about specific equipment to monitor its status and usage • As an Equipment Manager, I want the ability to update the details of current equipment to keep records up-todate • As an Equipment Manager, I want to be able to delete equipment from the system once it's no longer in use. • As an Equipment Manager, I want to track equipment performance and maintenance history through a dashboard.	As an Equipment Manager, I want the system to automatically notify me when equipment is due for maintenance or inspection
Supplier management (IT22168740)	• As a Supplier Manager, I want to add new suppliers to the system with their full details. • As a Supplier Manager, I want the ability to update the details of current suppliers. • As a Supplier Manager, I want to be able to delete a supplier from the system.	As a Supplier Manager, I want to track supplier performance via a dashboard.

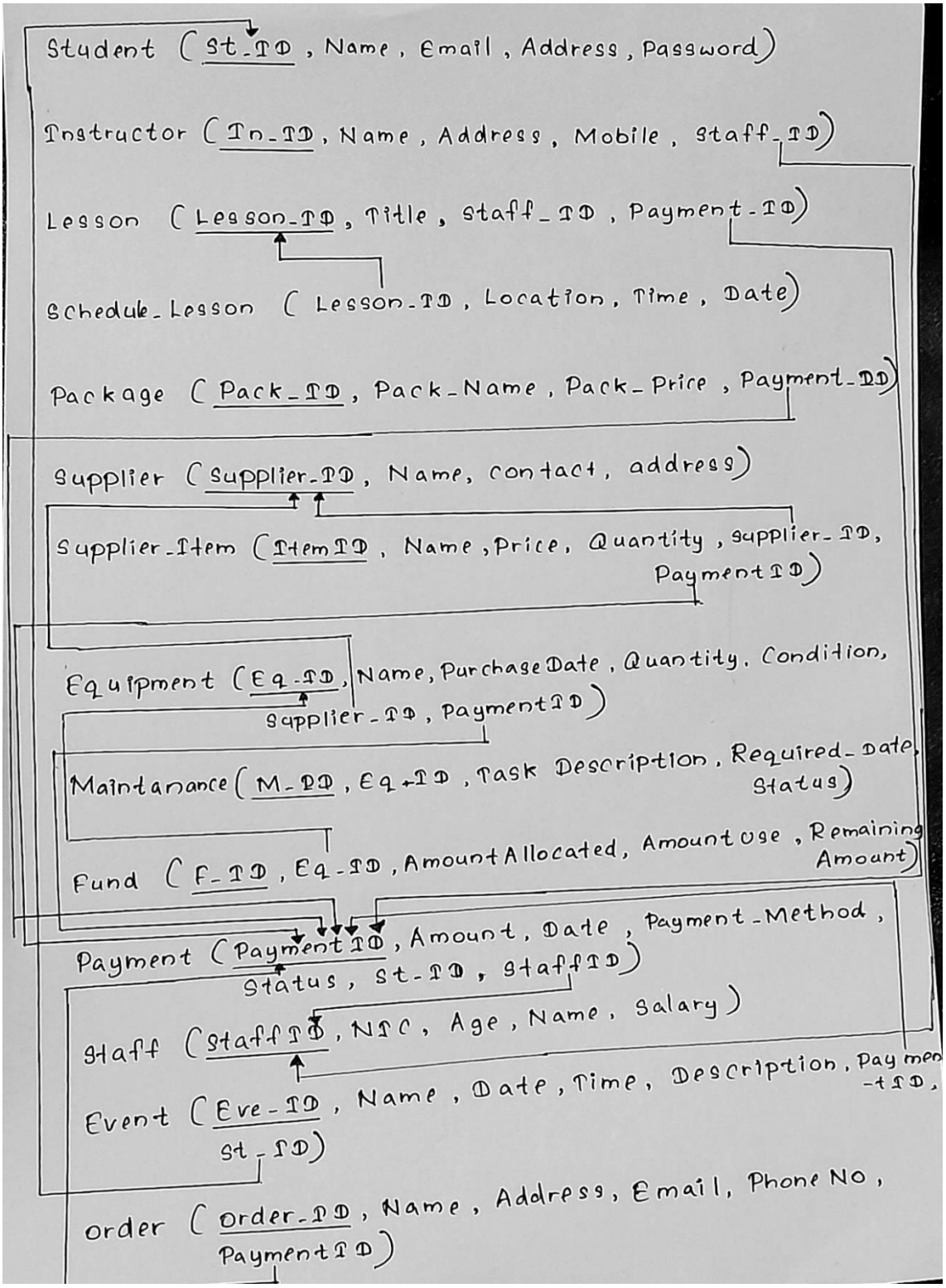
Sales & rental management IT22256300	• As a customer, I want to navigate to the surfing board page so that I can browse available surfboards. • As a customer, I want to select a surfing board from the list of displayed boards, allowing me to view its details before making a purchase. • As a customer, I want to click the "Order Now" button on the surfing board page so that I can initiate the purchase process • As a customer, I want to submit the order form by clicking the "Submit Order" button, confirming the transaction and completing the purchase process.	As a Customer, I want to place an order for a surfing board within the system after selecting it, so that I can complete the purchase
Staff Handling (IT22235688)	• As a Staff Manager , I need a System to Manage staff member details and register them. • As a Staff Manager, I must effectively search for staff member by their Id ensuring quick and efficient staff member identification. • As a Staff Manager , I want to delete staff member details if that Staff member no longer needed so that I can manage the system efficiently.	As a Staff Manager , I want to log in to the system (in completed due to not integrating function.
Student Management (IT22281500)	• As a student , I want the option to register using Google authentication • As a new student, I want to register for the surfing school by providing my personal details (name, email, phone number, and address), so that I can create an account and enroll in lessons. • As a student, I want to enroll in a specific surfing lesson, so that I can reserve my spot in the class and participate. • As an admin, I want to add, update, or remove students from the system, so that I can manage students effectively.	As an admin, I want to view reports on user registration statistics.

Repository Link - https://github.com/Pasindya/Surfing-Project-MERN-

02. ER Diagram



03. Normalized Schema



04.Test Case Design

Lesson Management

Test Cases for Adding New Lessons

- Dashboard: Log in as a instructor , navigate to Lesson management section, and add new lesson.
- Enter valid details like lesson title, date, time, location, and description.
- Add lesson and expect results: Scheduled lessons appear on user side.

Test Case 2: Checking Updated Lesson Details

- Dashboard: Log in as a instructor, navigate to Lesson management section, and select existing lesson.
- Update lesson details and expect results: Scheduled updated lessons appear on both user and instructor side.

Test Case 3: Generating Report About Scheduled Lessons

- Dashboard: Log in as a instructor navigate to Lesson management section, select lesson list, and download report.

Event Planning

Test Case 1: Successful Login

- Verifies a legitimate user can log in effectively.
- User enters a legitimate username and password.
- The system displays an error message if the username and password combination are incorrect.

Test Case 2: Incorrect Password and Username

- Checks if the system displays an error message if the username and password combination are incorrect.
- User remains on the login screen and receives an error message.

Test Case 3: Enter Data and Click "Login"

- Confirms that if the username or password field is empty, the system displays an error message.
- User enters credentials and logs in.

Test Case 4: Following Login, Look for Events

• Verifies that the user can search for events after logging in.

• User opens the event dashboard and uses event categories to find events.

Test Case 5: Event Registration
• Verifies that the user can register a event after logging in.
• User logs in with valid credentials and fills in the required fields.

Test Case 6: View Event
• Verifies that the user can view an existing event.

Test Case 7: Update Event
• Verifies that the event register details can update.
• User logs in with valid credentials and selects an event.

Test Case 8: Delete Event
• Verifies that the event register details can be deleted.

Test Case 9: Generate Report
• Verifies that the user can cancel an existing event.

Payment

• **Test Case 1**: Ensuring successful user payment for surf lessons, equipment, or events.
• **Test Case 2:** Accurate display of payment amount and type.
• **Test Case 3:** Correct calculation of payment amount based on offers and discounts.
• **Test Case 4:** User can update payment details before final submission. • **Test Case 5:** Payment updated in the system once user modifies personal details.

• Payment Process and Verification: Ensuring user can proceed to payment page after reviewing payment.
• **Test Case 7:** Correct payment total and details are shown on payment confirmation page.
• **Test Case 8**: Users can make payments using various methods.
• **Test Case 9:** Valid payment data results in clear error messages.
• **Test Case 10:** System handles successful payment processing and provides confirmation message.
• **Test Case 11:** System handles failed payment attempts and allows user to retry or choose another payment method.

• Payment Offers: Verifying special offers are applied correctly during payment processing.
• **Test Case 13:** Correct final total is displayed after applying offers. • **Test Case 14**: Users are notified if an offer code is invalid or expired.

• Payment History: Users can view a list of their previous payments. • **Test Case 15:** Users can export or download their payment history.

• Payment Management for Admin: Administrator can view all user payments, update payment status, search for specific payments, issue refunds, and update payment history.

Equipment maintenance

• Sign In: Verify successful customer sign-in with valid credentials.
• Add Equipment: Verify system navigates to add equipment form.
• Validate form data and pass data to database.
• Edit Equipment: Verify system edits existing equipment.
• Delete Equipment: Verify system deletes existing equipment.
• Search Equipment: Verify system searches for equipment by name or details.
• Report Generation: Verify system generates accurate, necessary equipment maintenance report.
• Low Stock Detection: Verify system notifies admin when stock levels are low.

Supplier management

• **Test Case 1**: Ensure user is directed to supplier form.
• **Test Case 2:** Confirm form details are correct.
• **Test Case 3**: System validates form data.
• **Test Case 4:** Data saved in database and new supplier appears in dashboard.
• **Test Case 5**: Existing supplier details can be edited.
• **Test Case 6:** Existing supplier can be deleted.
• **Test Case 7:** Suppliers can be searched by name.
• **Test Case 8:** Detailed report of supplier restock records can be generated.
• **Test Case 9:** Report contains accurate and complete information.
• **Test Case 10:** Report can be downloaded in PDF format.

Sales & rental management

Test Case 1: Navigating to Surfing Board Page
• Open website and navigate to home page.
• Click on "Surfing Boards" section.
• Result: Successful navigation to available surfboards.

Test Case 2: Selecting a Surfing Board
• Navigate to the board page.
• Select a board and view details.
• Result: Accurate display of board details.

Test Case 3: Ordering a Surfboard
• Click "Order Now" on surfboard details page.
• Fill out "Place Your Order" form with valid personal and payment details.
• Click "Submit Order" to send a success message.

Staff Handling

Test Cases for Staff Addition
• Successful Staff Addition: User logged in as admin.
• Test Steps: Navigate to "Add Staff" page, enter valid staff details.
• Test Case 2: Missing Required Fields: User left fields empty. • Test Case 3: Invalid Email Format: User entered invalid email format.

Test Cases for Viewing Staff Details
• View Staff List: User logged in as admin.
• Test Steps: Ensure staff details are displayed correctly.

• Test Case 5: Search Staff: User entered a staff member's name.

Test Cases for Updating Staff Information
• Successful Staff Update: User modifies staff details.
• Test Case 7: Invalid Phone Number Format: User enters invalid phone number.

Test Cases for Deleting a Staff Member
• Successful Staff Deletion: User selects a staff member to delete.
• Cancel Deletion: User cancels the deletion.

Test Cases for Staff Login
• Successful Login: User has valid credentials.
• Test Case 11: Incorrect Credentials: User enters incorrect credentials.

Test Cases for Role-Based Access
• Restricted Access for Non-Admin Users: User logged in as a staff member.

Student Management

Student Registration:
• Verifies successful registration with valid and required details.
• Tests for registration failures if required fields are blank or if invalid data is provided.
• Tests for sign-in failures if invalid credentials are provided or if an unregistered account is accessed.

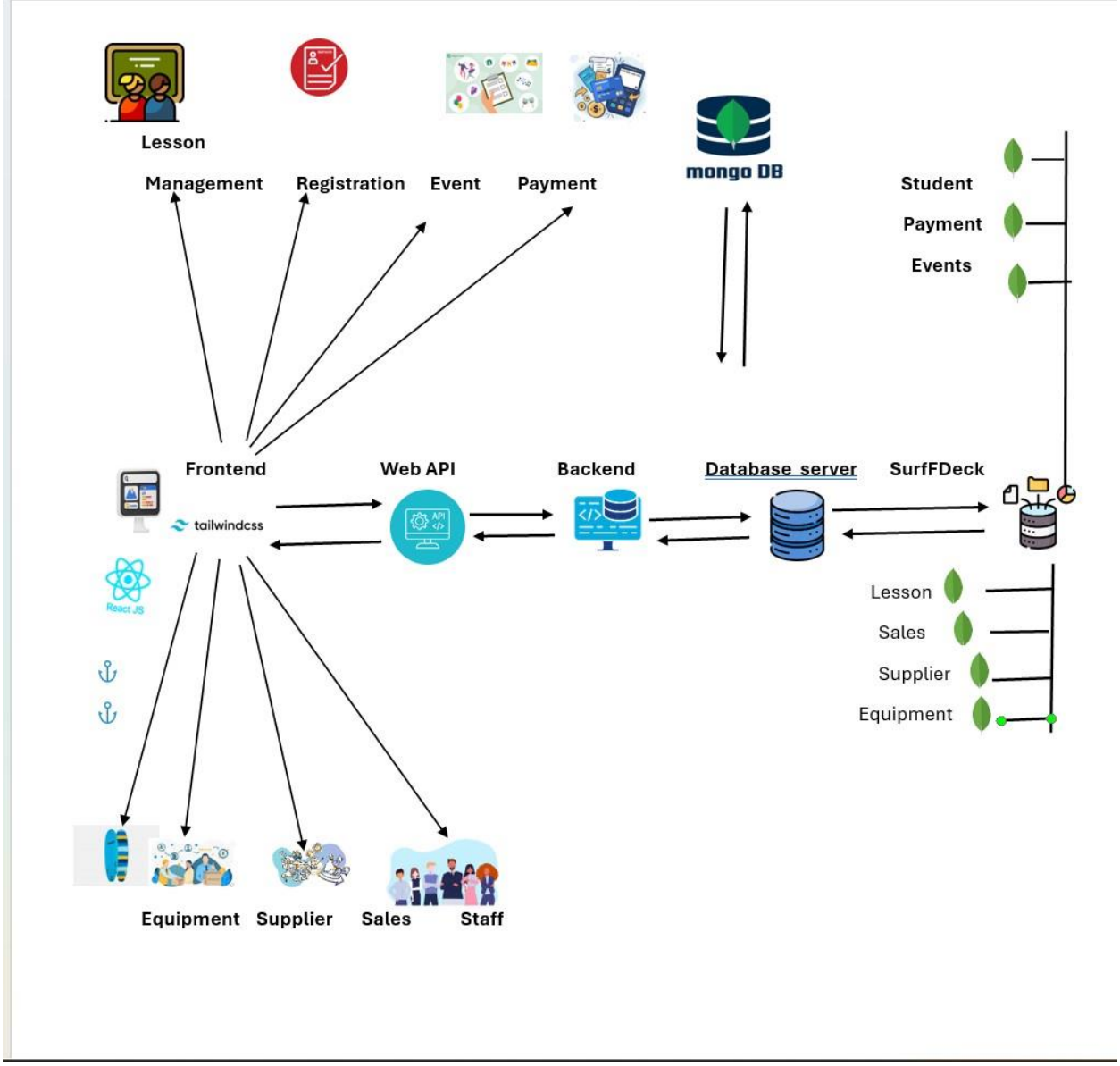
Tests for admin access:
• Verifies system owner's ability to grant admin access to selected users.
• Tests for admins' ability to view and search for specific users.
• Tests for admins' ability to remove users from the system. **Tests for report generation:**
• Verifies accurate and necessary user details report.
• Tests for report download in PDF format.

Performance tests:
• Verification of system's ability to handle concurrent registrations and sign-ins.
• Tests for password reset emails and secure user password storage.
• Tests for unauthorized access to admin functionalities and sensitive user data.

User interface tests:

• Verification of intuitive and easy-to-navigate interface.

05. High Level System Design



06. Innovative parts of the project

6.1.Lesson management

Instructors can create customizable lesson packages that cater to different skill levels, such as beginner, intermediate, or advanced. Students can select and purchase lesson packages that best fit their learning objectives, and the system tracks their progress through these packages.

6.2.Event planning

Students can easily register for upcoming events (surf camps, competitions, workshops) via the system's web and mobile platforms. They can view event details, select packages, and complete the registration process online.

6.3.Payment

Using search function implemented, admin can easily find relevant orders using orderId, customer name or userId.

Dynamic price changes such as users input codes or select discount options.

A graphical representation of discounts applied (e.g., showing price reductions over time or across product categories).

6.4.Equipment maintenance

Manager can add new equipment dynamically, ensuring all inputs (equipment name, quantity, date, etc.) are validated for accuracy (e.g., the date must be in the future, quantity must be a positive integer). This guarantees smooth and error-free equipment management.

6.5.Supplier

Admins can dynamically add suppliers to the system, ensuring all input data undergoes validation checks to maintain accuracy and data integrity

6.6.Sales & Rental

Implement a personalized surfboard recommendation system based on customer history and browsing habits, and introduce a real-time order tracking feature for live updates throughout the order process.

6.7.Staff handling

The staff handling system uses AI to allocate staff based on their expertise, availability, and past performance. This ensures that the right staff members are assigned to events or lessons where they can provide the best service.

6.8.Student management

he mobile app syncs with the web-based platform, giving students real-time access to their lesson schedules, progress reports, and feedback. It also supports push notifications for lesson updates, instructor messages, and reminders.

07. Commercialization

We are pleased to present our all-inclusive Serf Deck Surfing School Management System, a game-changing platform that will completely change the way surfing schools operate by combining vital features like reservation, lesson administration, safe payment processing, staff coordination, equipment tracking, and user engagement into a single, seamless package. Customers can easily schedule lessons and events using an intuitive interface, and instructors may use powerful lesson planning tools that are customised for different skill levels to deliver individualised training. By utilising reputable payment gateways, our secure payment processing system ensures a seamless transaction experience, protecting customers and schools from fraud. Enhancing student engagement and participation, the event management capabilities make it easier to plan and carry out surfing competitions, workshops, and community events. Tools for effective staff management facilitate communication... While our specialised equipment management system helps monitor inventory levels and schedule maintenance, ensuring that your surfing gear is always ready for use, efficient staff management tools streamline communication, shift scheduling, and performance tracking, ensuring your team stays motivated and in sync. Improved user profiles also let students keep track of their accomplishments and progress, which promotes a feeling of belonging and individual development. Our Surfing School Management System is the best tool to improve customer happiness, streamline operations, and spur growth—whether you run a major academy or a small local surf school. With it, you can concentrate on imparting the love of surfing to others and teaching them how to surf. Transform your operations now and join the management of the future for surfing schools!

8.Individual Contribution

8.1.Lesson Management (IT22127778)

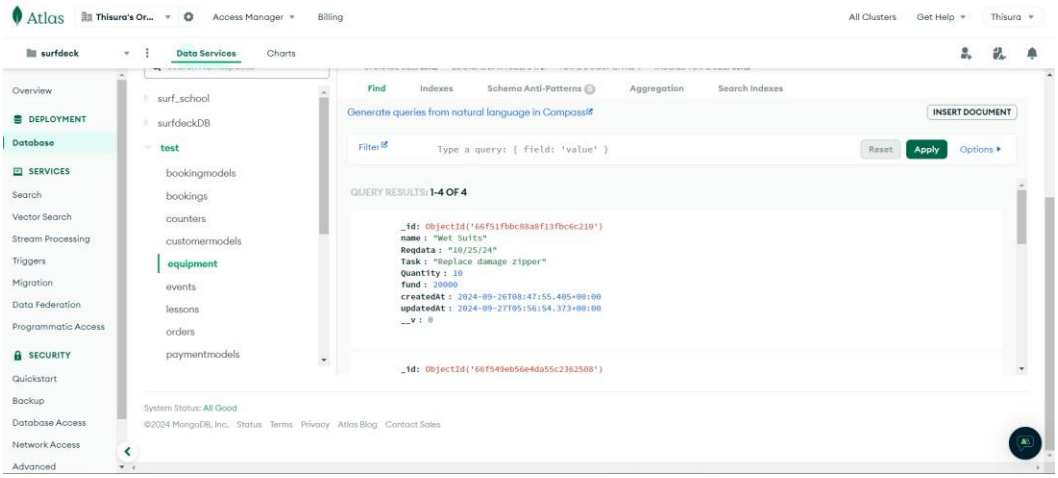
8.1.a The module working on

The Lesson Management Module streamlines lesson scheduling within the surfing school system, allowing instructors to add, update, view, and delete lessons. It ensures accuracy of lesson details and generates unique IDs for easy organization. The module is currently integrating lesson creation and database, benefiting both instructors and students.

8.1.b Completion level

Completed features	In completed features
Lesson scheduling form Delete unwanted lessons Search filter by lesson name Update lessons if there are any changes Generate report about scheduled lessons	○ Graphical display about scheduling lessons

8.1.c SQL Quries



8.1.dAlgorithm

Step1. Start the process.

Step2. Display the Add Lesson Form with required input fields (Lesson Name, Date, Time, etc.).

Step3. Wait for the instructor to enter the input

Step4. Validate the input:

• Check if all fields are filled.

• Validate the format of the date and time**.**

Step5. If validation fails, display an appropriate error message and repeat step 3.

Step6. If validation passes, create a Lesson Object with the instructor input.

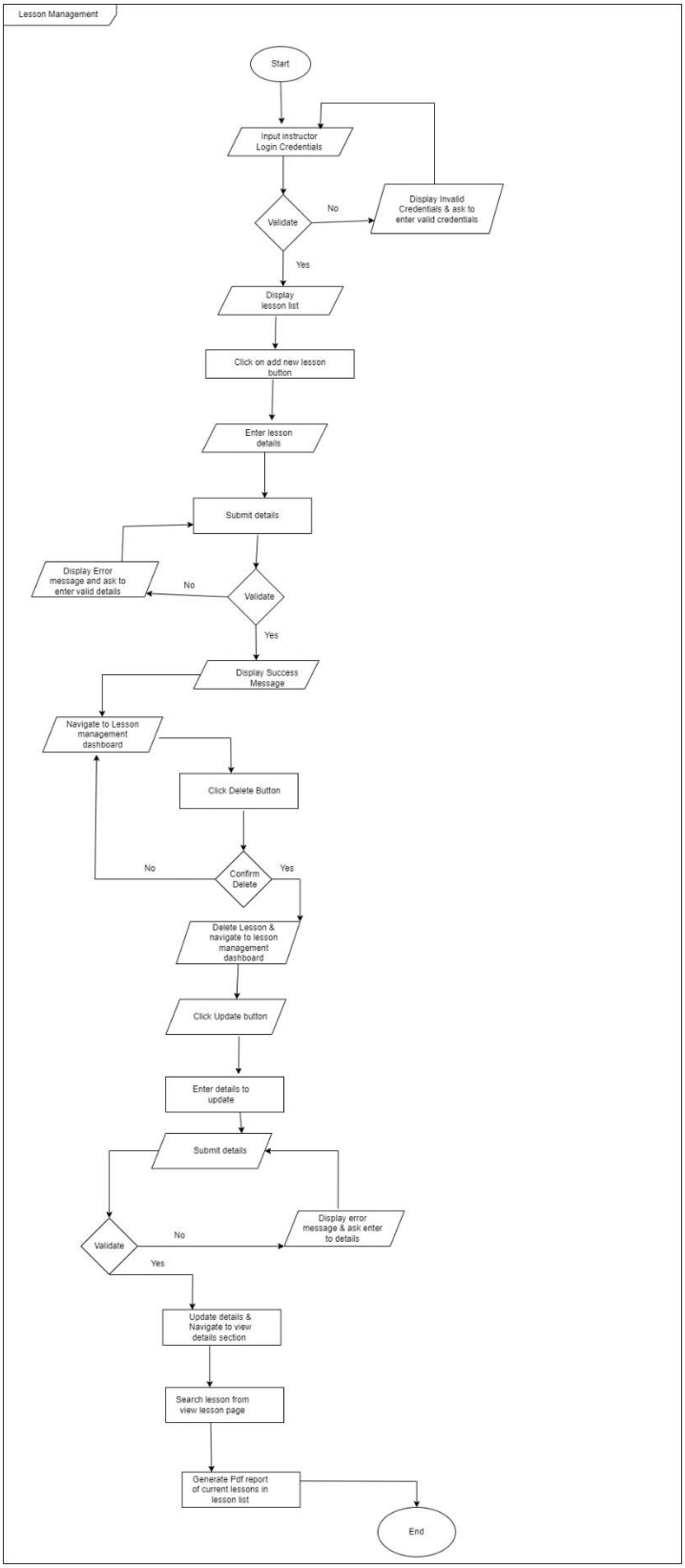
Step7. Assign a unique ID to the lesson.

Step8. Save the scheduled lesson in the database.

Step9. Display a success message & scheduled lessons under Lessons in user side as upcoming lessons .

Step10. End the process**.**

8.1.e Flow chart



8.1.f.Pseudocode for add lesson

START

FUNCTION displayAddLessonForm():

DISPLAY "Lesson Title"

DISPLAY "Lesson Date"

DISPLAY "Lesson Time"

DISPLAY "Lesson Location"

DISPLAY "Lesson Description"

DISPLAY "Submit Button"

FUNCTION validateLessonInput(title, date, time, location, description):

IF title IS EMPTY OR date IS EMPTY OR time IS EMPTY OR location IS EMPTY OR description IS EMPTY THEN

RETURN "Error: All fields are required."

IF date IS NOT A VALID DATE THEN

RETURN "Error: Invalid date format."

IF time IS NOT A VALID TIME THEN

RETURN "Error: Invalid time format."

RETURN "Success"

FUNCTION addLesson(title, date, time, location, description):

lessonID = GENERATE_UNIQUE_ID()

DATABASE INSERT INTO Lessons(lessonID, title, date, time, location, description)

VALUES (lessonID, title, date, time, location, description)

RETURN "Lesson has been added successfully."

FUNCTION handleLessonFormSubmission():

INPUT title = GET_INPUT("Lesson Title")

INPUT date = GET_INPUT("Lesson Date")

INPUT time = GET_INPUT("Lesson Time")

INPUT location = GET_INPUT("Lesson Location")

INPUT description = GET_INPUT("Lesson Description") validationResult = validateLessonInput(title, date, time, location, description) IF validationResult IS NOT "Success" THEN

DISPLAY validationResult RETURN addLessonResult = addLesson(title, date, time, location, description)

DISPLAY addLessonResult

REDIRECT TO "Lesson List Page"

END

8.2.Event planning(IT22243980)

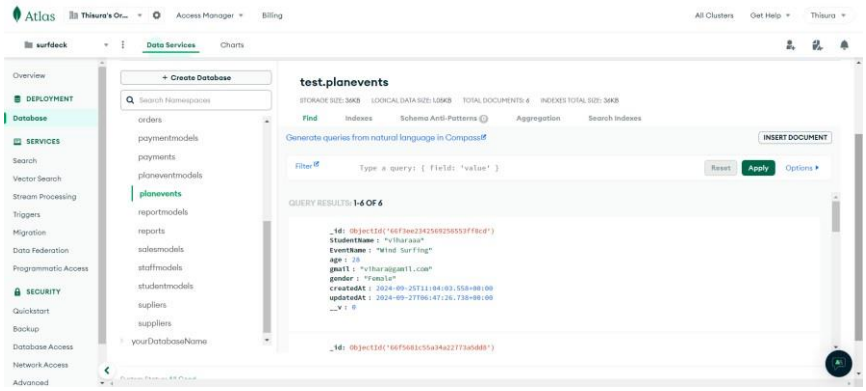
8.2.a. The module working on

Students can easily register for events by visiting the event dashboard, making payments, and filling out a registration form. The event manager can view, update, and delete registration details, search registered students, and generate daily registration files. They can also update notifications and upcoming events, making it a flexible and easy way for students to join surf events.

8.2.b . Completion level

Completed Features	In complete task
Login to the system Register to the events View event registered details Update event details Delete saved details Search register student's details Generate daily register details as a pdf	• Upload notifications and upcoming events • As student generate registration form as pdf

8.2.c. SQL Quries



8.2.d.Algorithm

Step1: The user login

1. Start
2. The login page with the password and username boxes should be displayed
3. User inputs their username and password. 4. Confirm: Verify the username and password correspond to the credentials that are saved.

    - If it is valid, move on to step 5.
    - If incorrect: Proceed to step 2 and display the error notice.

5. Display the Dashboard and save user session data.
6. Navigate to the Event Management System.

Step 2: Registering for the Event

1.The Event Registration Page will now be displayed.
2. Input: Name, date, time, location, and description of the event are entered by the user into the Event Form. 3. Send the form in:

- Verify the input fields to make sure necessary fields are filled in, the dates are valid, etc.

- Display an error message and go back to the form if validation is unsuccessful. • Save event data in the database if validation is successful.

  4. Show a confirmation message after successful event registration.

Step 3: Attend to Events

1. Show the Manage Events Page with an events history and forthcoming schedule.
2. An event can be chosen by the user to View Details, Edit, or Delete.

- View: Show specifics about the event.
- Edit: Complete the form with the event details and submit it.
- Delete: Verify and take the event out of the database.

Step 4: Notifications of Events

1. Input: The administrator plans out alerts for forthcoming occasions.
2. The system notifies registered participants via SMS or email at the designated time (e.g., one week, one day before to the event). 3. Show the success message for the notice.

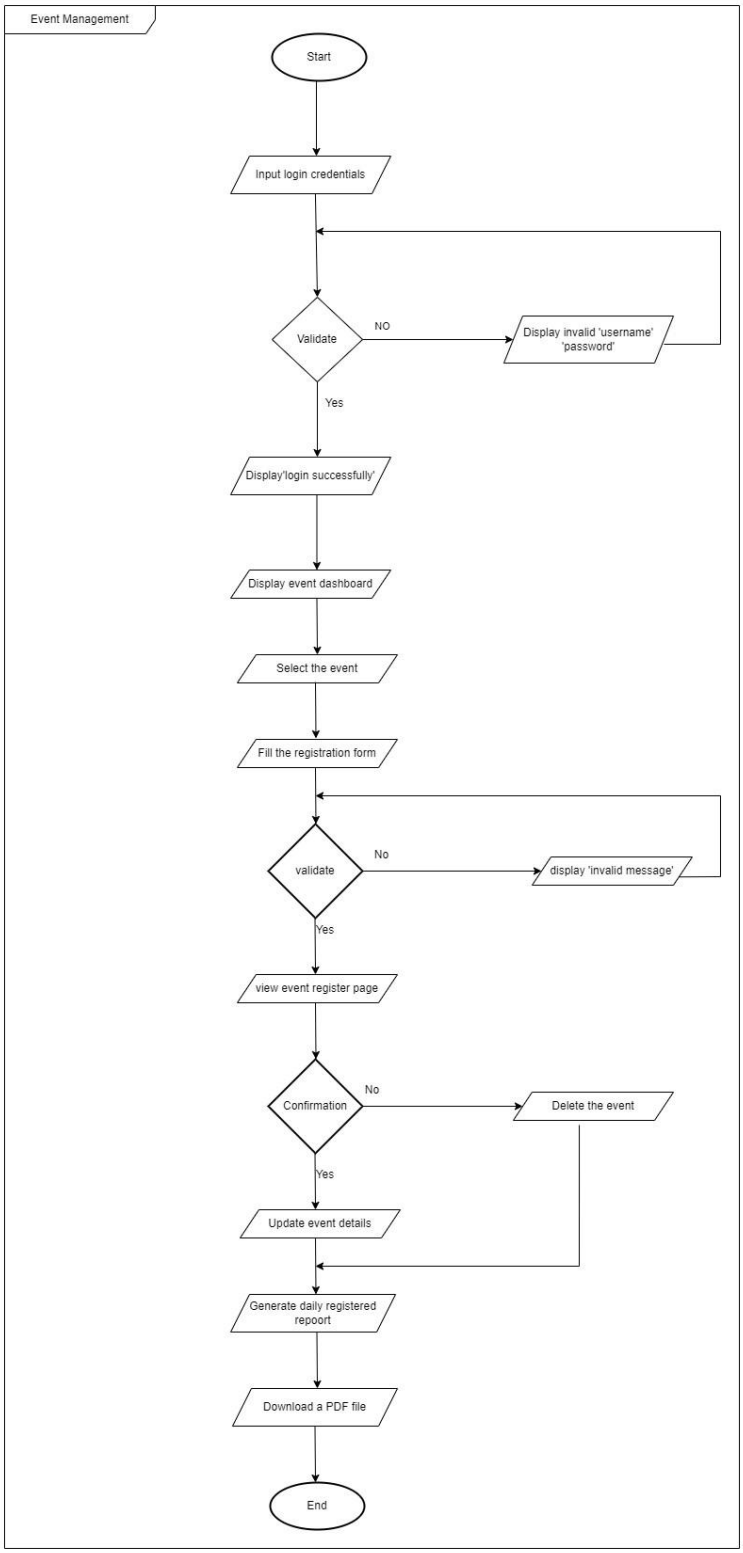
Step 5: Logging out as a user 1. A user selects the Logout option.

2. Put an end to the user session.

3.Take me back to the Login page.

4. End.

8.2.e.Flow chart



8.2.f.Pseudocode

```
FUNCTION eventPlanning()
DISPLAY "Event Planning Dashboard"
WHILE TRUE:
    DISPLAY "1. Create Event"
    DISPLAY "2. View Events"
    DISPLAY "3. Edit Event"
    DISPLAY "4. Delete Event"
    DISPLAY "5. Exit"
    INPUT userChoice
    IF userChoice == 1:
      CALL createEvent()
    ELSE IF userChoice == 2:
      CALL viewEvents()
    ELSE IF userChoice == 3:
      CALL editEvent()
    ELSE IF userChoice == 4:
      CALL deleteEvent()
    ELSE IF userChoice == 5:
      BREAK
    ELSE:
      DISPLAY "Invalid choice. Try again."
END FUNCTION
FUNCTION createEvent()
  DISPLAY "Enter Event Name:"
  INPUT eventName
```

```
    DISPLAY "Enter Event Date:"
    INPUT eventDate
    DISPLAY "Enter Event Time:"
    INPUT eventTime
    DISPLAY "Enter Event Location:"
    INPUT eventLocation
    DISPLAY "Enter Event Description:"
INPUT eventDescription          eventID =
GENERATE_UNIQUE_ID()
    SAVE event to database with eventID, eventName, eventDate, eventTime, eventLocation,
eventDescription
    DISPLAY "Event created successfully."
END FUNCTION
FUNCTION viewEvents()
    FETCH allEvents from database
IF allEvents IS EMPTY:
        DISPLAY "No events available."
ELSE:
        FOR EACH event IN allEvents:
            DISPLAY "Event ID:", event.eventID
            DISPLAY "Event Name:", event.eventName
            DISPLAY "Event Date:", event.eventDate
            DISPLAY "Event Time:", event.eventTime
            DISPLAY "Event Location:", event.eventLocation
            DISPLAY "Event Description:", event.eventDescription
            DISPLAY "-----------------------"
```

```
END FUNCTION
FUNCTION editEvent()
   DISPLAY "Enter Event ID to Edit:"      INPUT
eventID     event = FETCH event by eventID from
database     IF event IS FOUND:
       DISPLAY "Edit Event Name (current: ", event.eventName, "):"
       INPUT newEventName
       DISPLAY "Edit Event Date (current: ", event.eventDate, "):"
       INPUT newEventDate
       DISPLAY "Edit Event Time (current: ", event.eventTime, "):"
       INPUT newEventTime
       DISPLAY "Edit Event Location (current: ", event.eventLocation, "):"
       INPUT newEventLocation
       DISPLAY "Edit Event Description (current: ", event.eventDescription, "):"
       INPUT newEventDescription
       UPDATE event in database with newEventName, newEventDate,
newEventTime, newEventLocation, newEventDescription        DISPLAY "Event
updated successfully."
   ELSE:
       DISPLAY "Event not found."
END FUNCTION
FUNCTION deleteEvent()
   DISPLAY "Enter Event ID to Delete:"     INPUT
eventID     event = FETCH event by eventID from
database
   IF event IS FOUND:
```

DELETE event from database

DISPLAY "Event deleted successfully."

ELSE:

DISPLAY "Event not found."

END FUNCTION

8.3.Payment(IT22165848)

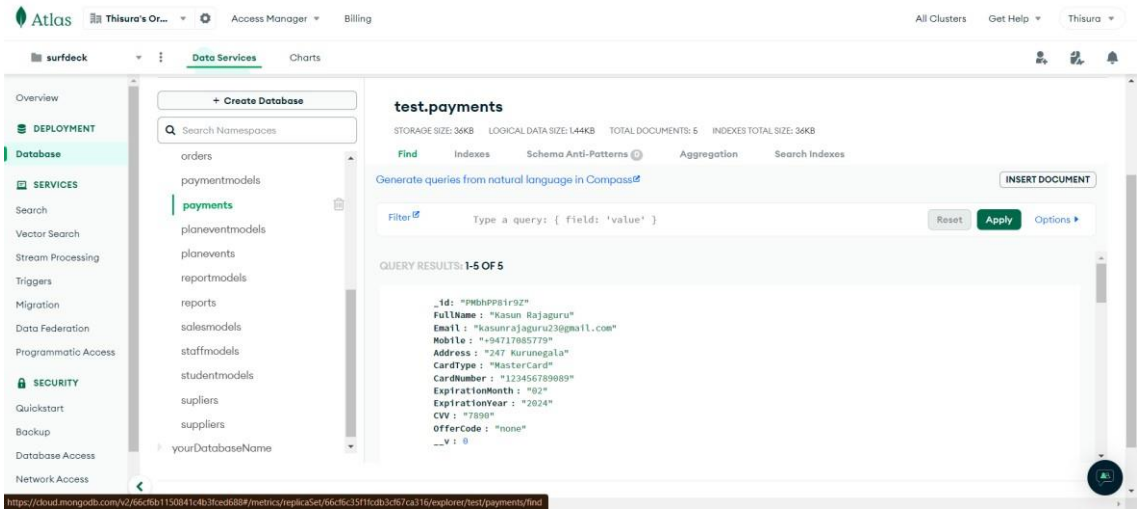
8.3.a. The module working on

This module of the Surfing School Management System enables administrators and users to effectively handle and process payments. While administrators can manage operations like processing discounts and creating payment reports, users can check payment information and transaction status. To provide efficient financial record administration, the module has features for viewing, updating, and deleting payment information. The administrator can create comprehensive payment reports in PDF format that include transaction specifics such as payment ID, client name, contact information, and exclusive deals. This feature simplifies corporate processes by monitoring revenue flow, usage of special offers, and payment history. Transparency is ensured by the user-friendly interface, which helps the surfing school's system run smoothly and expand.

8.3.b . Completion level

Completed Features	In complete task
Form to enter personal details and card details. View saved personal details and deleted or updated personal details only. View payment summary, generate report and send it as a pdf	• Applying selected offers to payments.

8.3.c. SQL Quries



8.3.d.Algorithm

Step 1: Start

Step 2: Fetch all payment records from the server.

- If the fetch is successful, proceed to Step 3.
- If there is an error in fetching, display "Error fetching payments" and retry fetching.

Step 3: Display a dropdown list containing all payment IDs.

- The user selects a payment ID.

Step 4: Handle payment selection.

- When a payment ID is selected:
    - Check if the payment ID exists in the list of payments.
    - If the payment ID exists:
        - Proceed to Step 5.
    - If the payment ID does not exist:
        - Display "Invalid payment ID" and go back to Step 3.

Step 5: Display selected payment details (Full Name, Email, Mobile, Address).

- If the user does not select a payment ID, prompt them to choose one from the dropdown list.
- Once a valid payment ID is selected, proceed to Step 6.

Step 6: User clicks on the "Generate PDF" button to create a payment report.

Step 7: Check if a payment is selected.

- If no payment is selected: ○ Display "Please select a payment" and go back to Step 4.
- If payment is selected:
    - Proceed to Step 8.

Step 8: Generate a PDF report.

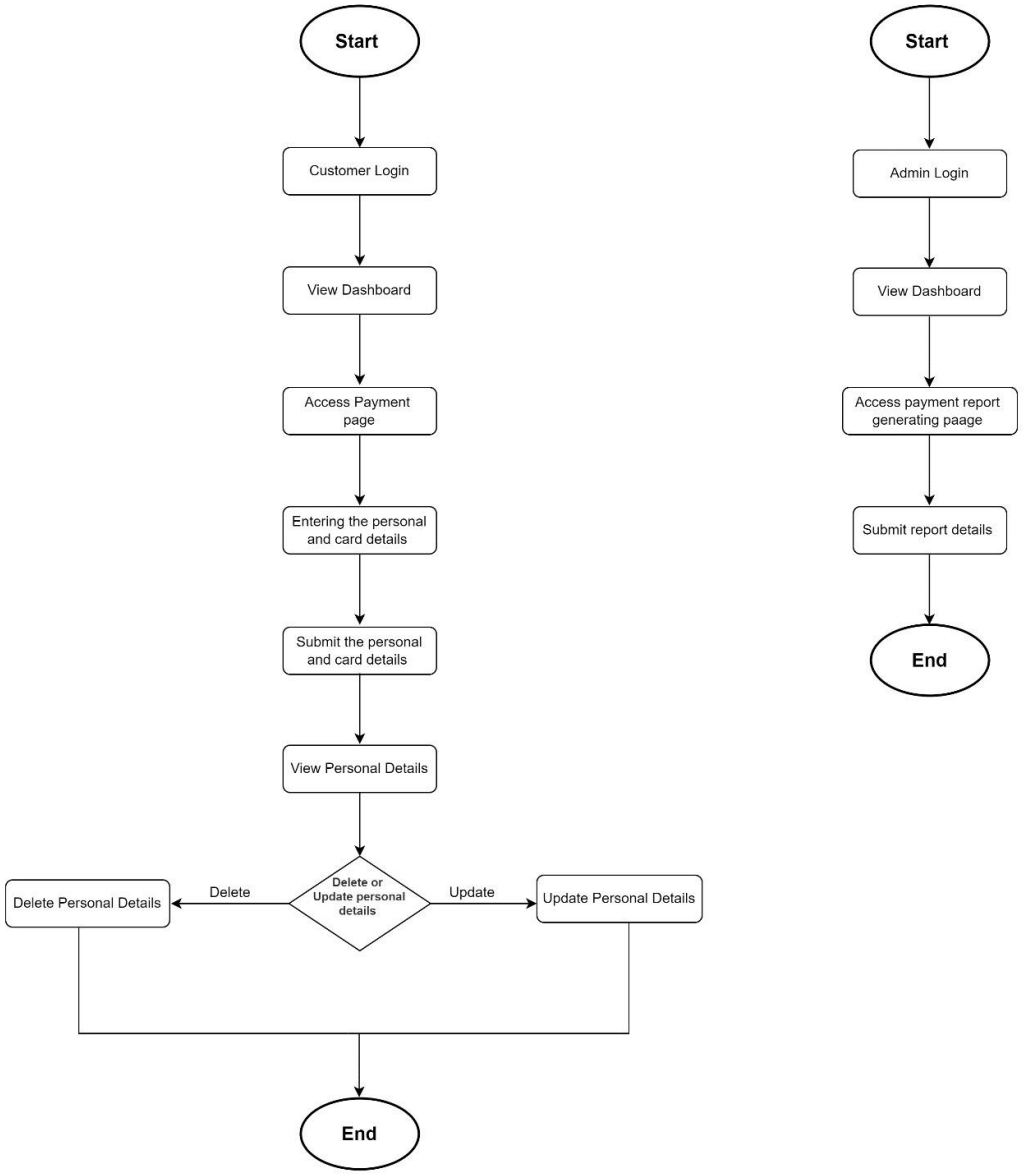
- Create a new PDF document.
- Add background color and the company logo.
- Add company name, address, and current date.
- Add the title "Payment Report" to the center of the page.
- Display "Payment Successful" in green text at the center.
- Add payment details (Payment ID, Full Name, Email, Mobile, Address) centered on the page.

Step 9: Save the PDF file.

- Save the PDF with the filename Payment Report_{PaymentID}.pdf.

Step 10: End.

8.3.e.Flow chart



8.3.f.Pseudocode

START SurfingSchoolManagementSystem

IMPORT necessary libraries and components (useState, useEffect, jsPDF, Headernav, Footer, logo image, etc.)

FUNCTION SurfingSchoolManagementSystem

// State Initialization

INITIALIZE payments as an empty array

INITIALIZE selectedPaymentId as an empty string

INITIALIZE selectedPayment as null

INITIALIZE specialOffers as an empty array

INITIALIZE selectedOffer as an empty string

INITIALIZE reportData as null

// Fetch Payment Data

USE useEffect to fetch payment data

DEFINE async function fetchPayments

TRY

SEND GET request to 'http://localhost:5009/payments'

CONVERT response to JSON

UPDATE payments with fetched data

CATCH error

LOG error to console

CALL fetchPayments()

// Fetch Special Offers Data

USE useEffect to fetch special offers

DEFINE async function fetchOffers

TRY

SEND GET request to 'http://localhost:5009/offers'

CONVERT response to JSON

```
      UPDATE specialOffers with fetched data
    CATCH error
      LOG error to console
  CALL fetchOffers()
// Handle Payment Selection
FUNCTION handlePaymentChange(event)
  SET selectedPaymentId to the value from event target
  FIND the payment matching selectedPaymentId from payments array
  SET selectedPayment to the matching payment
// Handle Offer Selection
FUNCTION handleOfferChange(event)
  SET selectedOffer to value from event target
  APPLY discount based on selected offer
// Generate PDF Report
FUNCTION generatePDF
  IF selectedPayment is null THEN RETURN
  INITIALIZE new jsPDF instance
  // Set background and logo
  SET background color to light blue (RGB 204, 255, 230)
  DRAW rectangle to fill entire page
  ADD logo to the document (top-left corner)
  // Add company name, address, and current date
  SET font size and ADD 'SurfDeck' and address
  GET current date and ADD it to the document
  // Add "Payment Report" title and "Payment Successful" message
  ADD title 'Payment Report'
```

```
  SET green text color for 'Payment Successful'
  CENTER and ADD the message

  // Add payment details (Payment ID, Full Name, Email, etc.)
  RESET text color to black
  FOR each detail in selectedPayment
    CENTER and ADD the detail
  // Save the PDF report
  SAVE PDF with filename `Payment_Report_{selectedPayment._id}.pdf`
// Render UI Components
RETURN
  RENDER main container with background image
  CALL Headernav component
  // Report Generation Section
  RENDER section for "Report Generation"
    RENDER select dropdown for Payment IDs
      ON change, CALL handlePaymentChange
    IF selectedPayment exists THEN
      RENDER selected payment details (Full Name, Email, etc.)
      RENDER "Generate PDF Report" button
        ON click, CALL generatePDF
  // Special Offers Section
  RENDER section for "Special Offers"
    RENDER select dropdown for special offers
      ON change, CALL handleOfferChange
  // Payment Summary Section
```

IF selectedPayment or selectedOffer exists THEN

RENDER summary of selected payment and applied offers

CALL Footer component

END SurfingSchoolManagementSystem

8.4Equipment & Maintenance(IT22083814)

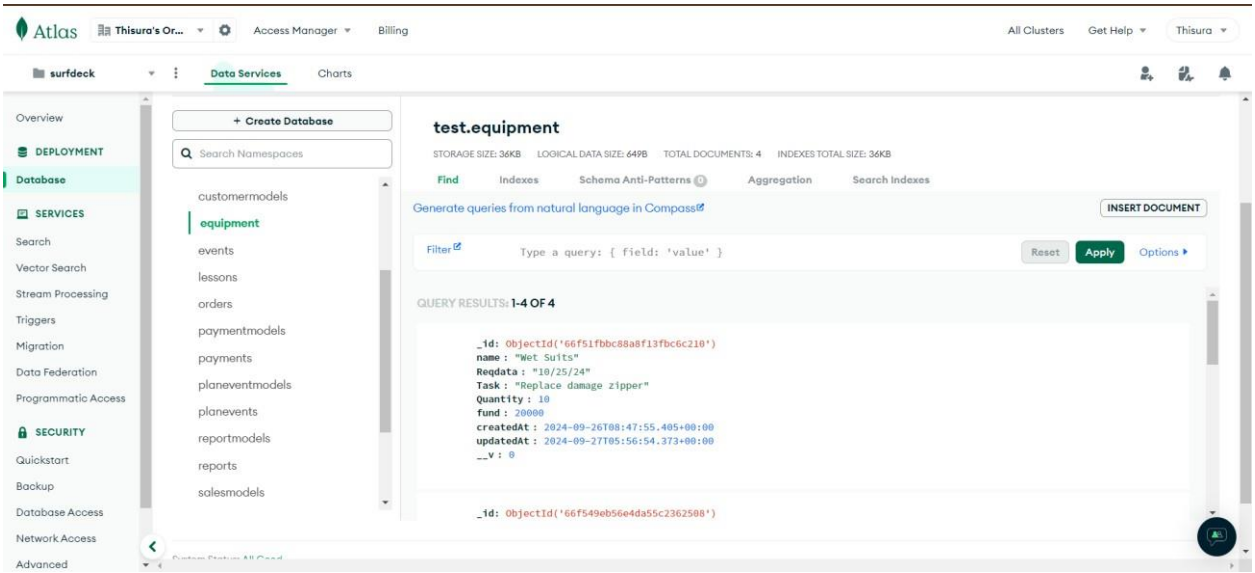
8.4.a. The module working on

The **Equipment Maintenance System** is designed to efficiently manage and track the maintenance of equipment, submit new maintenance requests, retrieve equipment details, and provide user-friendly access for both staff and administrators. It includes features such as creating maintenance requests, scheduling tasks, tracking repairs, and removing or updating equipment details after repairs. Additionally, it offers automatic notifications for upcoming maintenance or urgent equipment repairs and generates monthly maintenance reports

8.4.b . Completion level

Completed Features	In complete task
Adding New Equipment Maintenance Requests Editing and Updating Equipment Maintenance Tasks Removing Equipment from the Maintenance Queue Retrieving Equipment Details and Maintenance Status Generating Monthly Equipment Maintenance Reports	• Adding validations • Adding logo to report

8.4.c. SQL Quries



8.4.d.Algorithm

Step 1. Start

Step 2. Input login credentials (Username and Password)

Step 3. Validate the login credentials:

- a. If the credentials are valid:

- Display the home page.

- b. If the credentials are invalid:

- Display an error message “Invalid credentials.”

- Go back to Step 2.

Step 4. Navigate to the home page.

Step 5. Click on the profile picture.

Step 6. Navigate to the profile page.

Step 7. Click on the Equipment Maintenance section.

Step 8. Navigate to the Equipment Maintenance dashboard.

Step 9. Click the 'Add Maintenance Request' button.

Step 10. Navigate to the add maintenance request form.

Step 11. Fill out the form with the required details (e.g., equipment name, type of maintenance, required date, etc.).

Step 12. Click the ‘Submit’ button on the form.

Step 13. Validate the form data:

- a. If all fields are completed:

- Submit the data to the database.

- Display the request in the list of pending maintenance tasks.

- b. If not all fields are completed:

- Display an error message: “All fields are required.”

- Go back to Step 11.

Step 14. Navigate back to the dashboard if you want to edit existing requests.

Step 15. Click the equipment from the maintenance list to view details.

Step 16. Click the 'Edit' button to modify an existing maintenance request.

Step 17. Navigate to the maintenance request form with existing data.

Step 18. Edit the fields with necessary changes and click 'Save'.

Step 19. Click the 'Delete' button if you want to remove a maintenance request.

Step 20. Confirm the deletion of the maintenance request.

Step 21. Click the 'Search Equipment' option.

Step 22. Search maintenance requests by equipment name or maintenance task.

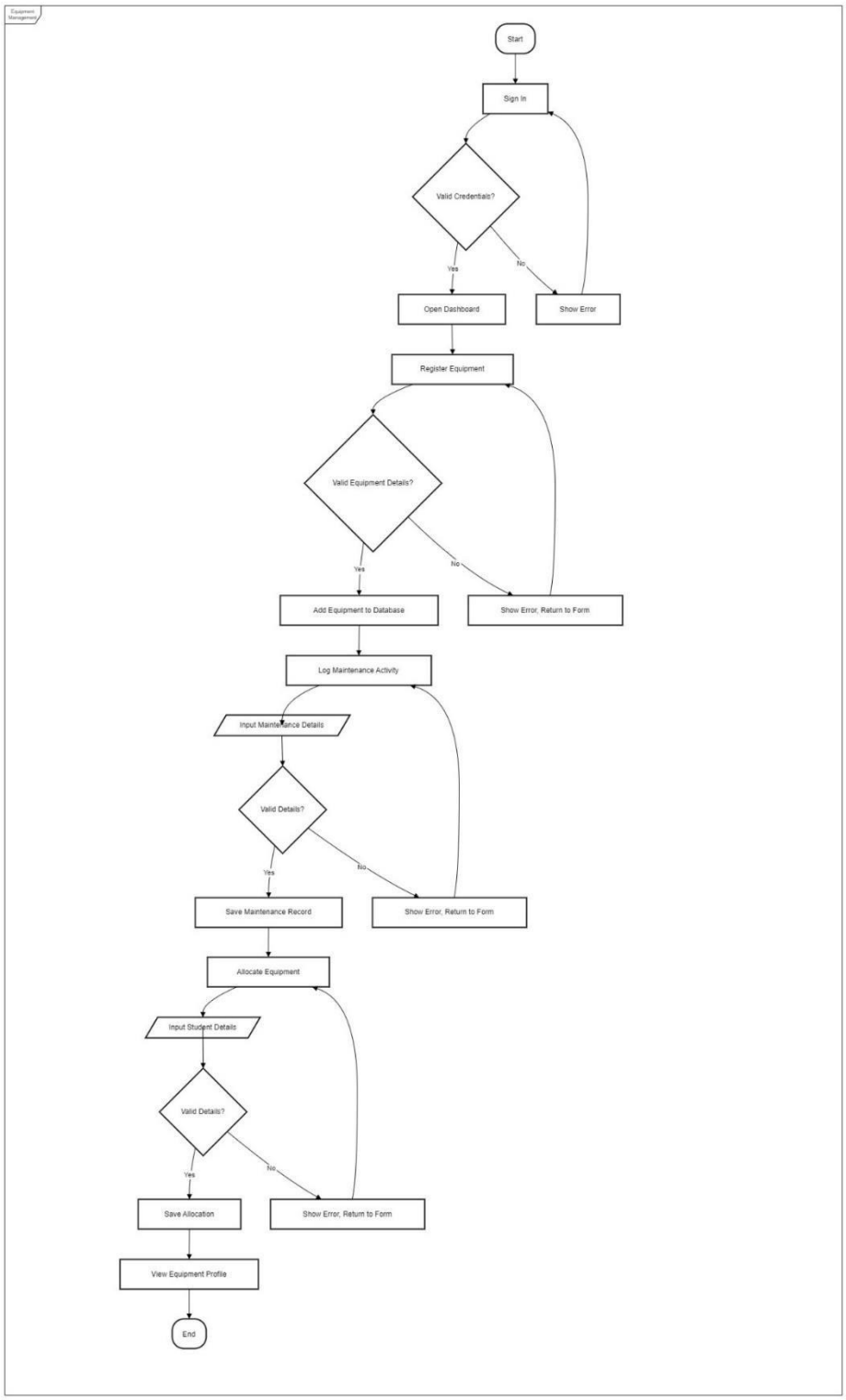
Step 23. Click the 'Generate Report' button.

Step 24. Generate a monthly equipment maintenance report in PDF format.

Step 25. Manager signs out of the system.

Step 26. End.

8.4.e.Flow chart



8.4.f.Pseudocode

BEGIN

Step 1: Start

Step 2: Input login credentials (Username and Password)

a. Input username and password.

Step 3: Validate the login credentials:

a. If credentials are valid: - Display the home page.

b. If credentials are invalid:

- Display an error message "Invalid credentials."

- Go back to Step 2.

Step 4: Navigate to the home page.

Step 5: Click on the "Profile" picture.

Step 6: Navigate to the profile page.

Step 7: Click the "Equipment Maintenance" button.

Step 8: Navigate to the Equipment Maintenance dashboard.

Step 9: Click the "Add Maintenance Request" button.

Step 10: Navigate to the add maintenance request form.

Step 11: Fill the form with maintenance request details (e.g., equipment name, task, required date).

Step 12: Click the "Submit" button on the form.

Step 13: Validate the form data:

a. If all data fields are completed:

- Pass the data to the database.

- Display the request in the maintenance list.

b. If data fields are incomplete:

- Display an error message "All fields are required." - Go back to Step 11.

Step 14: Go back to the Equipment Maintenance dashboard.

Step 15: Click on an existing maintenance request from the list to view details.

Step 16: Click the "Edit" button if you want to modify an existing maintenance request.

Step 17: Navigate to the request editing form with pre-filled data.

Step 18: Edit the fields with necessary changes and click "Save."

Step 19: Click the "Delete" button if you want to remove a maintenance request.

Step 20: Confirm the deletion of the request.

Step 21: Search for maintenance requests by equipment name or task.

Step 22: Click the "Generate Report" button.

Step 23: Generate a monthly equipment maintenance report as a PDF.

Step 24: Manager signs out from the system.

Step 25: End

END

8.5.Supplier Management(IT22168740)

8.5.a. The module working on

Supplier Management is a comprehensive system designed to manage suppliers, add new suppliers, retrieve suppliers, provide user-friendly supplier details access.

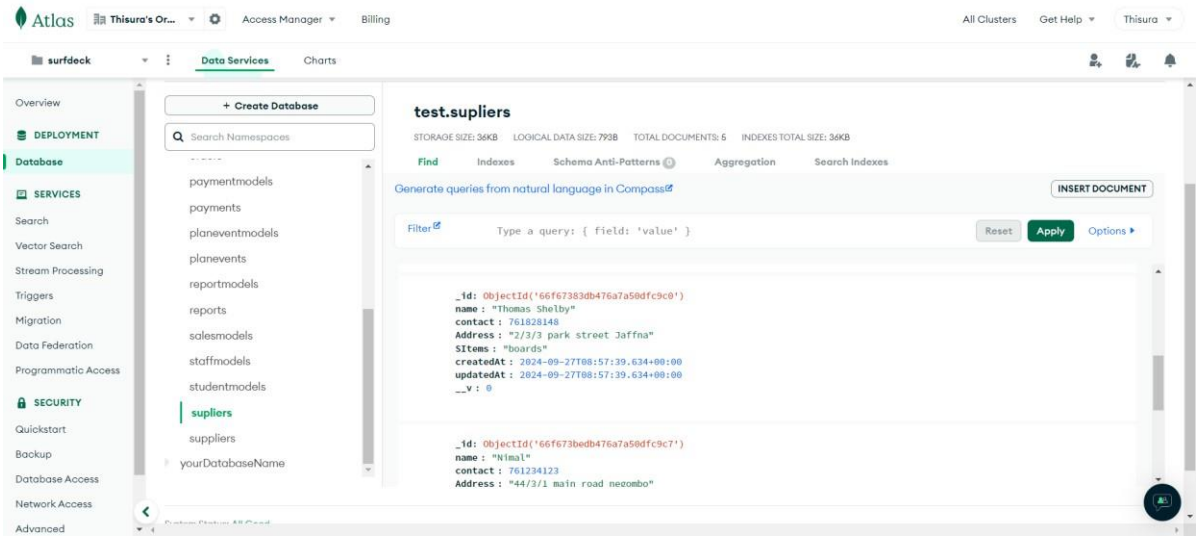
The system includes tasks for adding new suppliers, retrieving supplier details and ensuring userfriendly supplier details access. It also includes features for updating supplier details and remove suppliers.

. Lastly, Supplier Management generates report for suppliers' management purposes. The system includes tasks for designing a report template, developing backend logic to gather supplier data, and exporting the report in a printable format. The system aims to provide a user-friendly and efficient supplier management solution for businesses**.**

8.5.b . Completion level

Completed Features	In complete task
○ Add new suppliers to admin dashboard. ○ Edit supplier information form (Supplier Manager) ○ Remove suppliers ○ Generate report for supplier's records	• Add logo to report • Add validation

8.5.c. SQL Quries



8.5.d.Algorithm

Step 1: Start

Step 2: Input login credentials (Username and Password)

Step 3: Validate the login credentials:

a. If the credentials are valid:

• Display "Login successful" •

proceed to step 5

b. If the credentials are not valid:

• Display "Invalid Username and password" • Go

back to step 2 to input login credentials again

Step 4: Add new suppliers:
a. Provide the necessary details for supplier registration.
b. Validate the entered data.
c. If the data is valid, save it in the database.

Step 5: Navigate to the supplier details dashboard:

- Upon successfully adding a supplier, navigate to the homepage/dashboard.

Step 6: View all users and admins registered in the system.

Step 7: Update supplier details:

a. Provide the updated details for the supplier.
b. Validate the modified data.
c. If valid, update the supplier information in the database.

Step 8: Search for suppliers by supplier name.

Step 9: Send an email to the supplier for communication purposes.

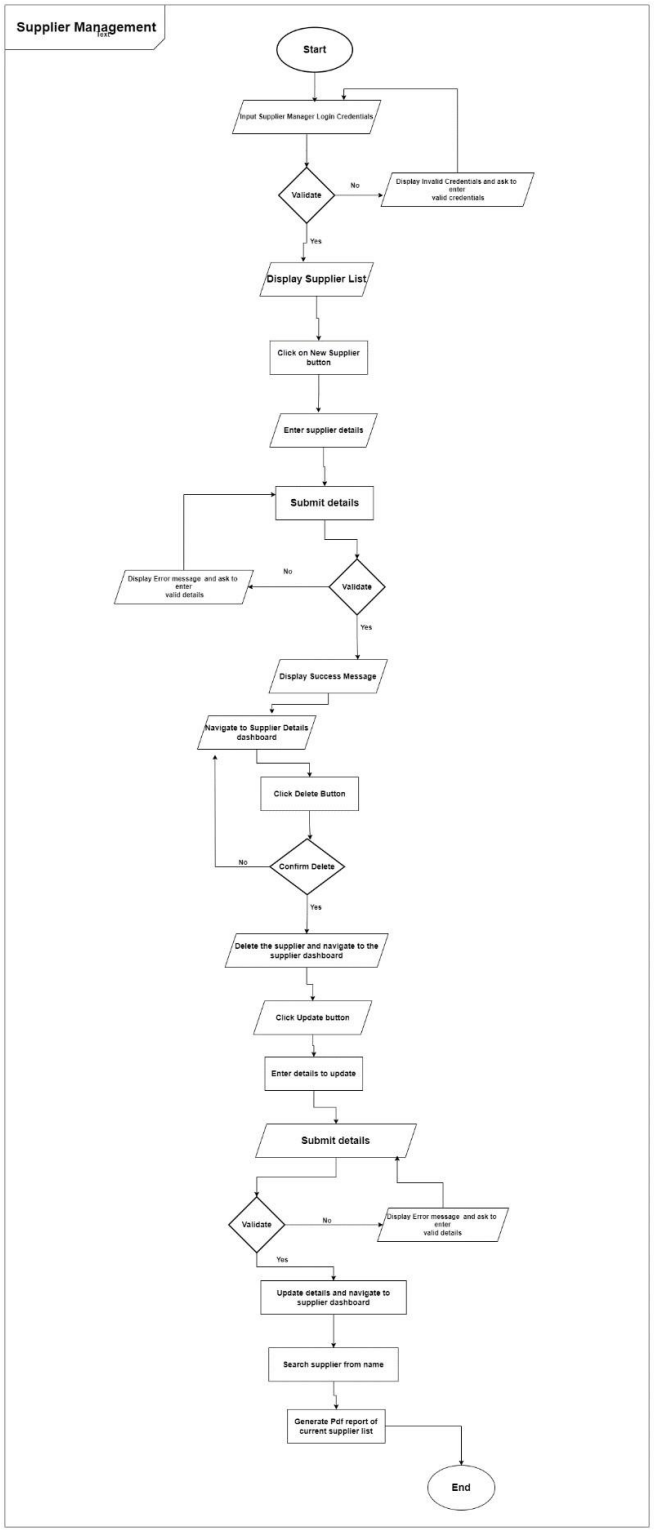
Step 10: Remove a supplier from the system.

- Ensure proper confirmation to avoid accidental deletion.

Step 11: Generate a report of supplier restock records and download restock invoice details in PDF format.

Step 12: End

8.5.e.Flow chart



8.5.f.Pseudocode

BEGIN

// Login Process

READ login credentials (Username and Password)

IF credentials are valid: // Validate credentials

DISPLAY "Login successful"

Proceed to Supplier details dashboard

ELSE

DISPLAY "Invalid Username and password"

Go back to Input login credentials

// Admin Dashboard

DISPLAY Admin dashboard

// Step 1: Add suppliers to admin dashboard

IF supplier details are valid: // Add suppliers

Save supplier to dashboard

View suppliers in supplier dashboard

Search suppliers by name

ELSE

DISPLAY "Error message"

// Step 2: Update supplier details

DISPLAY "Submit Update details"

IF updated details are valid: // Update suppliers

Update supplier details

// Step 3: Remove supplier

REMOVE supplier from dashboard

END

8.6Sales & Rental(IT22256300)

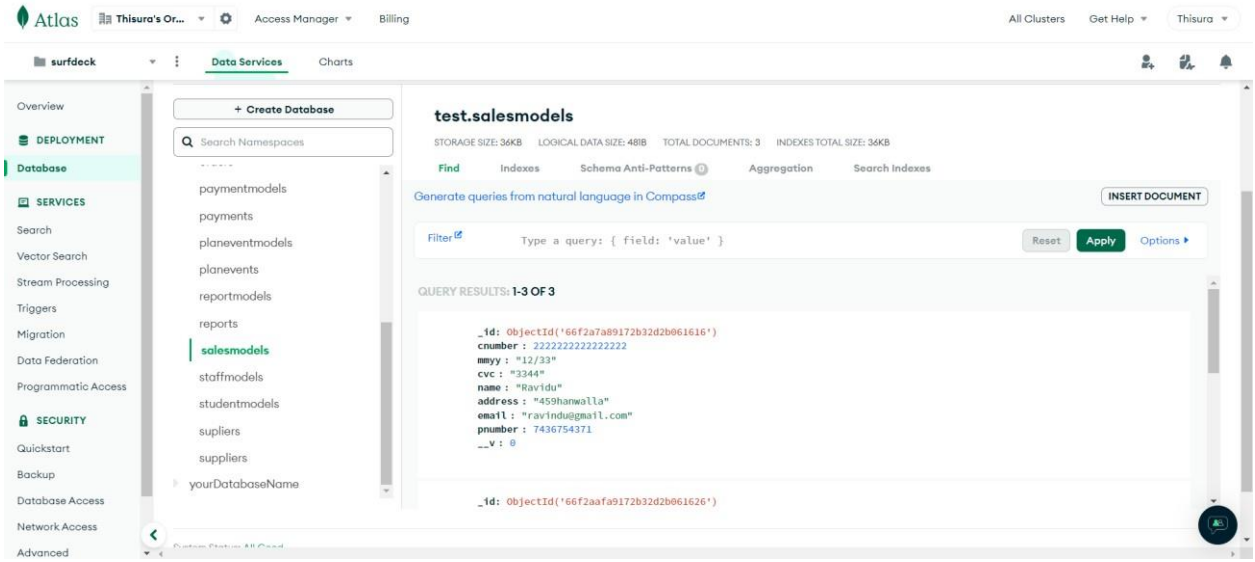
8.6.a. The module working on

The Surfboard Purchase and Order Management module in the Surfing Board Management System offers a seamless shopping experience for customers. It allows customers to browse available surfboards, make informed decisions, and place orders securely using an integrated payment gateway. This module enhances customer satisfaction by offering flexibility in payment methods.

8.6.b . Completion level

Completed Features	In complete task
Surfboard Selection Page Order Now Button Place Your Order Form Submit Order Functionality: Order Confirmation Message	• Generate report • Search function

8.6.c. SQL Quries

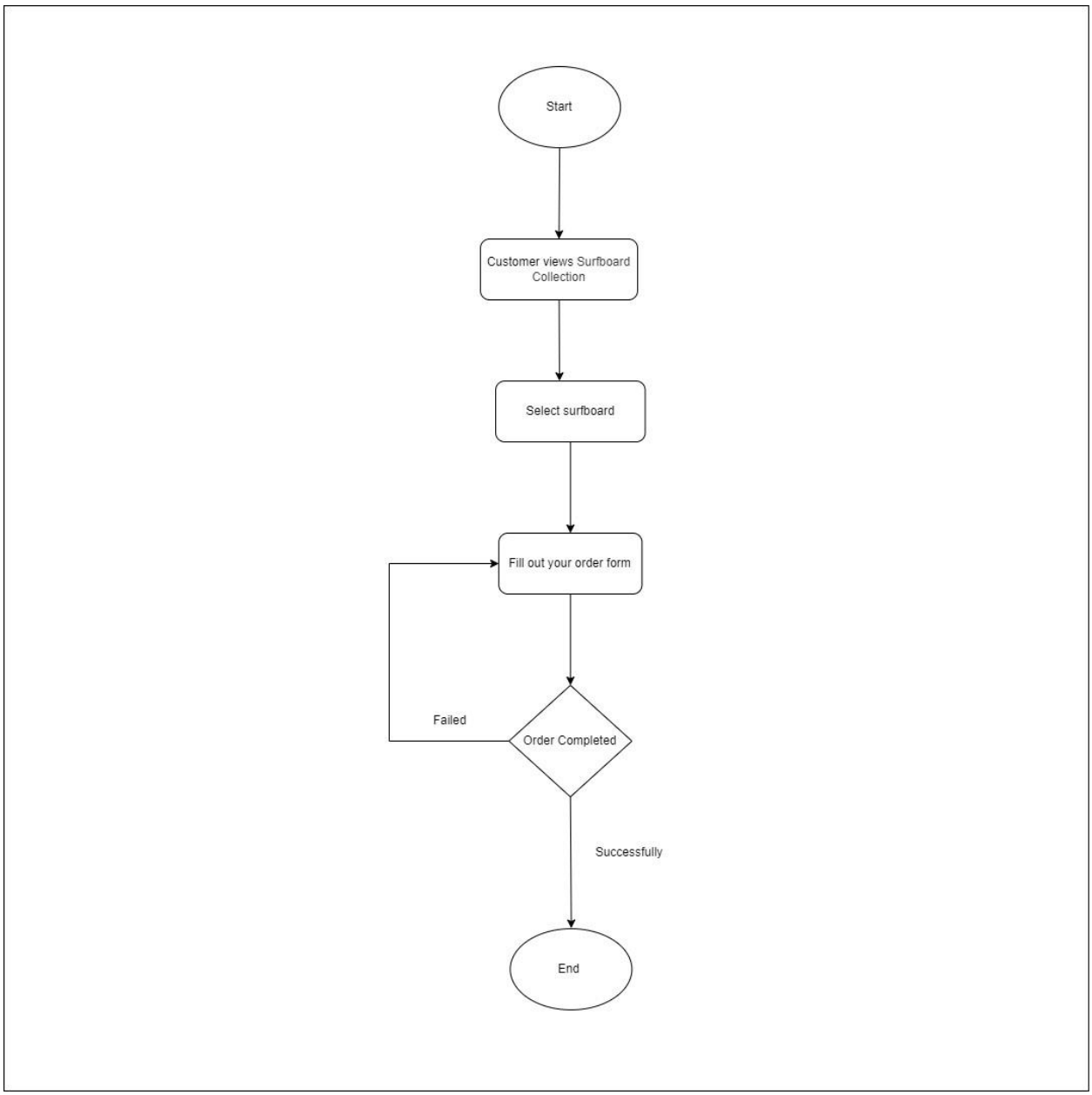


8.6.d.Algorithm

1. **Start**
2. Input customer details:
    - Card Number ○ Month & Year (MM/YY) ○ CVV ○ Name ○ Address ○ Email ○ Phone Number
3. **Validate input data**:
    - Check for non-empty fields. ○ Ensure the Card Number is valid.
    - Validate the Month & Year format (MM/YY).
    - Verify the CVV is a 3-digit number. ○ Ensure Email is in the correct format.
4. **If any field is invalid**:
    - Display "Invalid input. Please correct the fields." ○ Go back to Step 2 to re-enter the information.
5. **If all fields are valid**:
    - Proceed to Step 6.
6. **Submit order details**:
    - Send the entered details to the server for processing.

7. **Process order**:
    - If the order details are processed successfully:
        - Proceed to Step 8.
    - If there is an issue during processing:
        - Display "Order submission failed. Please try again."
        - Go back to Step 2 to re-enter details.
8. **Display success message**:
    - "Order placed successfully."
9. **End**

8.6.e.Flow chart



8.6.f.Pseudocode

```
FUNCTION PlaceOrderForm()

    DECLARE navigate AS useNavigate() // Get navigation function from react-router-dom

    DECLARE form AS Form.useForm() // Define form instance for handling the order form

    FUNCTION onFinish(values)

        LOG "Order form submitted with values" and print the values
```

```
    TRY
      SEND POST request to '/api/OrderRoute/PlaceOrder' with the order details (values)
      IF response.data.success is true
        DISPLAY success message from response.data.message using toast
        NAVIGATE to '/orderConfirmation' // Redirect user to the confirmation page
ELSE
        DISPLAY error message from response.data.message using toast
      END IF
    CATCH error
      DISPLAY "Something went wrong" using toast
    END TRY
  END FUNCTION
  RETURN div element WITH className="order-form-container" CONTAINING:
    div element WITH className="order-form" CONTAINING:
      h3 element WITH className="form-title" and text "Place Your Order"
Form element WITH layout='vertical', form=form, and onFinish=onFinish:
div element WITH className="form-row" CONTAINING:          div
element WITH className="form-item" CONTAINING:
            Form.Item element WITH label='Card Number' and
name='cardNumber':          Input element WITH placeholder='Enter your
card number'          div element WITH className="form-item" CONTAINING:
            Form.Item element WITH label='Month & Year (MM/YY)' and
name='expiryDate':
              Input element WITH placeholder='MM/YY'
div element WITH className="form-item" CONTAINING:
```

Form.Item element WITH label='CVV' and name='cvv':

Input element WITH placeholder='Enter CVV' div element

WITH className="form-item" CONTAINING:

Form.Item element WITH label='Name' and

name='name': Input element WITH placeholder='Enter

your name' div element WITH className="form-item"

CONTAINING:

Form.Item element WITH label='Address' and

name='address': Input element WITH placeholder='Enter

your address' div element WITH className="form-item"

CONTAINING:

Form.Item element WITH label='Email' and

name='email': Input element WITH placeholder='Enter

your email' div element WITH className="form-item"

CONTAINING:

Form.Item element WITH label='Phone Number' and

name='phoneNumber': Input element WITH placeholder='Enter your

phone number' div element WITH className="button-container"

CONTAINING:

Button element WITH className='submit-button', htmlType='submit', and text "Submit Order"

END FUNCTION

8.7Staff Handling(IT22235688)

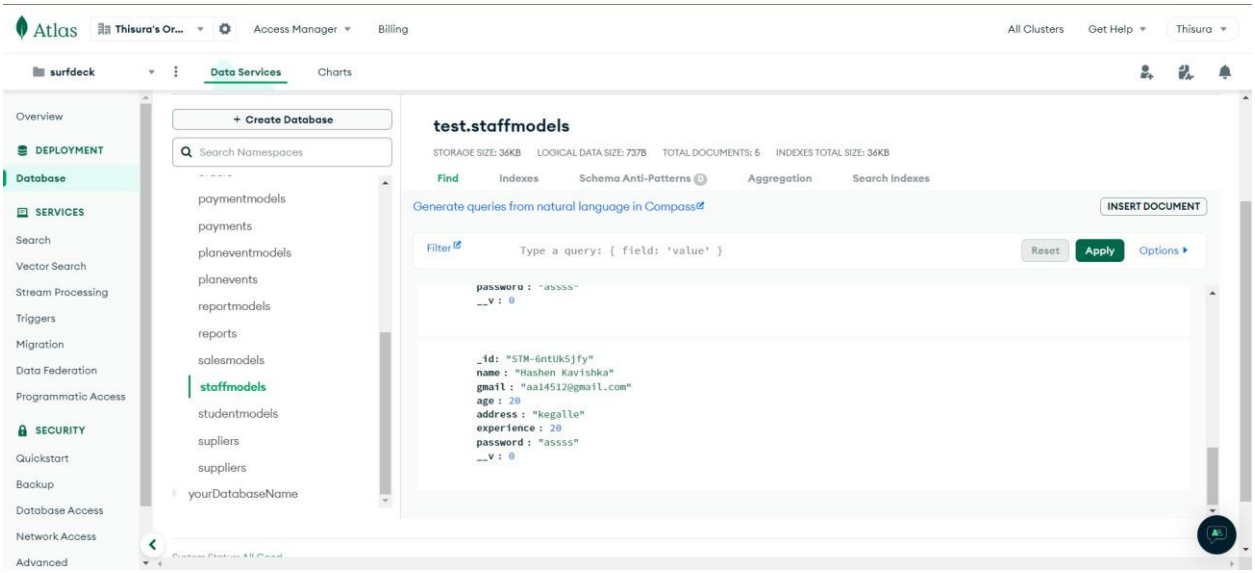
8.7.a. The module working on

The **Staff Handling Function** in the surf school management system is designed to streamline the management and coordination of staff members, including instructors and administrators. It encompasses a variety of features aimed at improving operational efficiency and ensuring smooth staff workflows. The module includes **Staff Profile Management**, which allows administrators to add, update, and manage staff details such as qualifications, roles, and assignments.

8.7.b . Completion level

Completed Features	In complete task
Staff Member registration form Updating Staff member information if needed. Delete Unwanted Staff members Search filter by Staff member ID Download Staff member detail in report	• Login functionality

8.7.c. SQL Quries



8.7.d.Algorithm

Algorithm for Staff Management Functions

1. **Staff Login** ○ Start ○ Prompt user for credentials (username and password).
    - Validate credentials:
        - If valid, proceed to Staff Dashboard.
        - If invalid, display an error message and prompt to retry.
    - End
2. **Add Staff Process** ○ Start Add Staff ○ Display form for staff details (name, position, etc.). ○ Get input from user for each field.
    - Validate input:
        - If all fields are valid:
        - Register staff in the system.
        - Display success message.

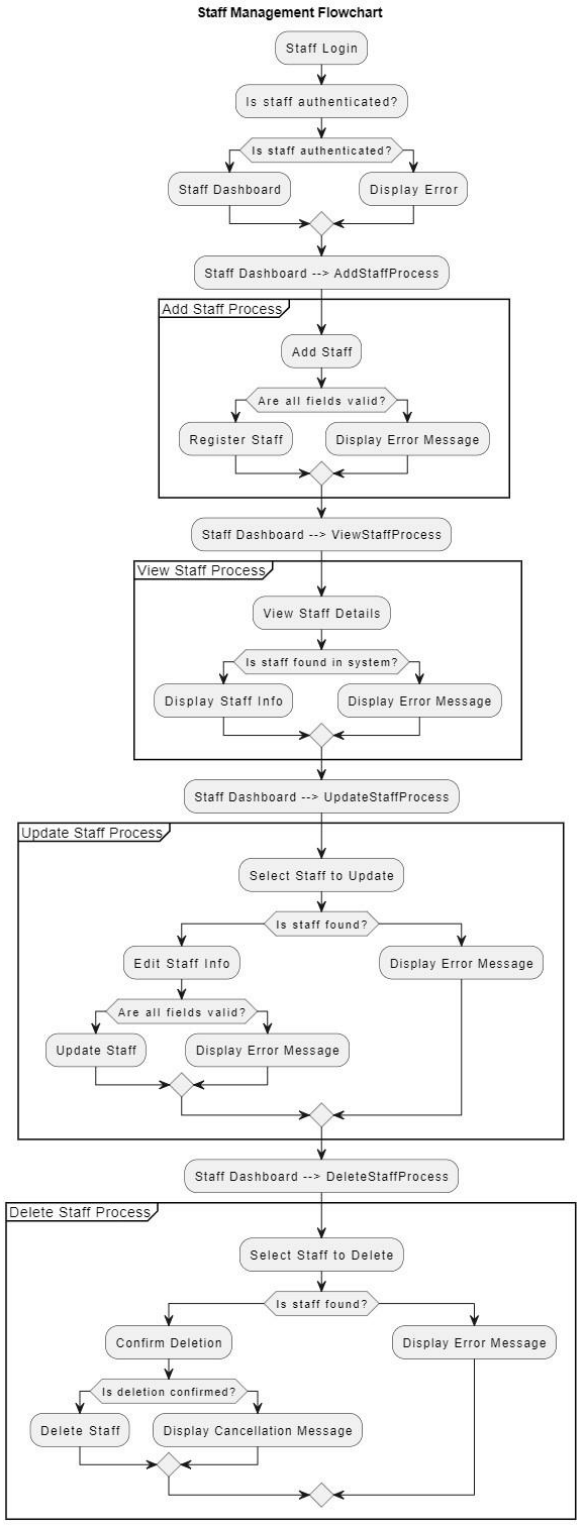
- If any field is invalid:
- Display an error message specifying which fields are incorrect.

- End Add Staff

3. **View Staff Process** ○ Start View Staff ○ Prompt user to enter staff ID or name.
   - Check if staff exists in the system:
     - If found, display staff information.
     - If not found, display an error message indicating that the staff member does not exist.
   - End View Staff
4. **Update Staff Process**
   - Start Update Staff ○ Prompt user to select staff member to update.
   - Check if staff exists:
     - If found:
     - Display current staff information.
     - Get new input for each field to update.
     - Validate input:
     - If all fields are valid:
     - Update staff information in the system.
     - Display success message.
     - If any field is invalid:
     - Display an error message specifying which fields are incorrect.
     - If not found, display an error message indicating that the staff member does not exist.
   - End Update Staff
5. **Delete Staff Process** ○ Start Delete Staff ○ Prompt user to select staff member to delete.

- Check if staff exists:
  - If found:
  - Ask for confirmation to delete.
  - If confirmed, delete staff from the system.
  - Display success message.
  - If not confirmed, display cancellation message.
  - If not found, display an error message indicating that the staff member does not exist.
- End Delete Staff

8.7.e.Flow chart



8.7.f.Pseudocode

```
START

  FUNCTION displayRegistrationForm():

    DISPLAY "Staff Name"

    DISPLAY "Email"

    DISPLAY "Phone Number"

    DISPLAY "Job Title"

    DISPLAY "Password"

    DISPLAY "Confirm Password"

    DISPLAY "Submit Button"

  FUNCTION validateInput(name, email, phone, jobTitle, password, confirmPassword):

    IF name IS EMPTY OR email IS EMPTY OR phone IS EMPTY OR jobTitle IS EMPTY OR
password IS EMPTY OR confirmPassword IS EMPTY THEN

      RETURN "Error: All fields are required."

    IF email DOES NOT CONTAIN '@' OR email DOES NOT CONTAIN '.' THEN

      RETURN "Error: Invalid email format."


    IF LENGTH(phone) IS NOT 10 THEN

      RETURN "Error: Phone number must be 10 digits long."


    IF LENGTH(password) < 6 THEN

      RETURN "Error: Password must be at least 6 characters long."


    IF password IS NOT EQUAL TO confirmPassword THEN

      RETURN "Error: Passwords do not match."


    RETURN "Success"
```

```
  FUNCTION registerStaff(name, email, phone, jobTitle, password):
    DATABASE INSERT INTO Staff(name, email, phone, jobTitle, passwordHash)
      VALUES (name, email, phone, jobTitle, HASH(password))
    RETURN "Staff registration successful."
FUNCTION handleFormSubmission():
    INPUT name = GET_INPUT("Staff Name")
    INPUT email = GET_INPUT("Email")
    INPUT phone = GET_INPUT("Phone Number")
    INPUT jobTitle = GET_INPUT("Job Title")
    INPUT password = GET_INPUT("Password")
    INPUT confirmPassword = GET_INPUT("Confirm Password")
    validationResult = validateInput(name, email, phone, jobTitle, password, confirmPassword)
    IF validationResult IS NOT "Success" THEN
      DISPLAY  validationResult            RETURN        registrationResult  =
registerStaff(name, email, phone, jobTitle, password)
    DISPLAY registrationResult
END
```

8.8.Registration(IT22281500)

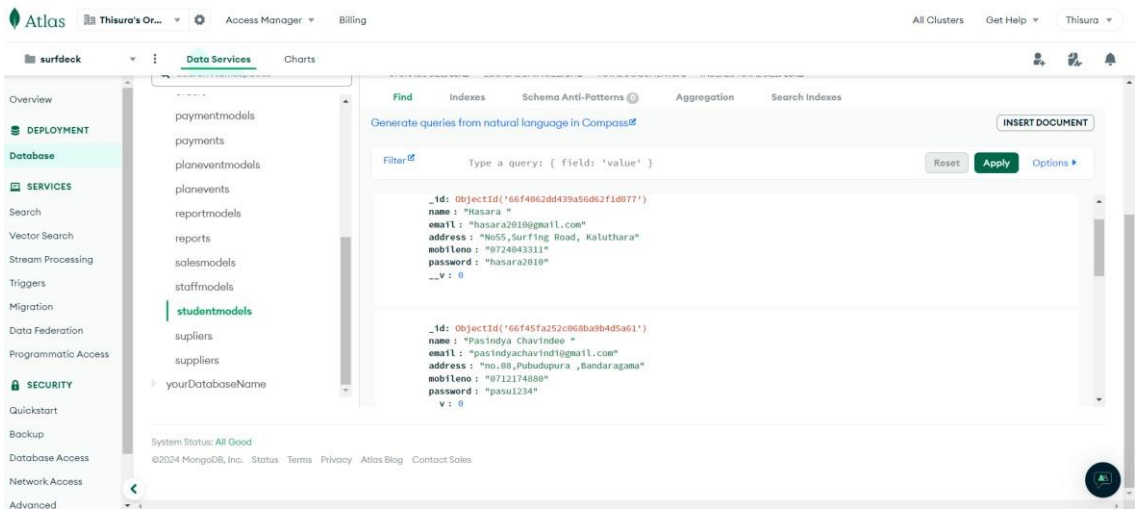
8.8.a. The module working on

The Customer Management System is a module that streamlines customer interaction, allowing registration, authentication, profile management, and administrative functions. It allows customers to register with Google authentication, manage their profile, and reset passwords securely. Administrators can view, search, and remove users, generate reports, and download user details in PDF format for system management and analysis.

8.8.b . Completion level

Completed Features	In complete features
Registration form User authentication Password reset	User profile

8.8.c. SQL Quries



8.8.d.Algorithm

Step 1: Start

Step 2: Customer Registration

a. Provide necessary details for registration.

b. Validate given data.

c. If data is valid, store it in the database.

d. Alternatively, allow registration using Google authentication.

Step 3: Sign In

a. Input login credentials (email and password)

b. Validate the credentials:

i. If valid, proceed to step 5.

ii. If invalid, display "Invalid email or password" and go back to step

Step 5: Homepage

a. Upon successful sign-in, navigate to the homepage.

Step 6: User Profile

a. Click on the user profile icon in the header.

b. Navigate to user profile.

c. Update user details including profile photo, username, email, password, address, and mobile number. d. Validate updated details:

i. If valid, update profile and display success message ii. If invalid, display warning message and prompt to give data in correct format. Step 7: Delete Account

a. Click on the delete button.

b. Validate if current user is the owner of the account.

c. If validated, delete the customer account.

Step 8: User Sign out.

a. Destroy session.

Step 9: Admin Access

a. Owner provides admin access using the database.

Step 10: View all users and admins in the system.

Step 11: Search users by username

Step 12: Remove users from the system.

Step13: Generate a report and download user details into a PDF document.

Step 14: Admin Delete Account

a. Admin clicks on the delete button for a user account.

b. Validate if admin has the authority to delete the account.

c. If validated, delete the user account. Step 15: Admin Sign out from the system. e. Destroy session.

Step 16: End

8.8.f.Pseudocode

```
function registerUser():
   // Step 1: Collect user input
   INPUT username
   INPUT email
   INPUT password
   INPUT confirmPassword
   // Step 2: Validate input
   IF username is empty OR email is empty OR password is empty OR confirmPassword is empty THEN
      DISPLAY "All fields are required"
      RETURN
   IF password does not match confirmPassword THEN
```

```
        DISPLAY "Passwords do not match"
        RETURN

    // Step 3: Validate email format
    IF email is not in valid format THEN
        DISPLAY "Invalid email address"
        RETURN
    // Step 4: Check if user already exists
    IF user with the given email or username already exists in the database THEN
        DISPLAY "User with this email or username already exists"
        RETURN
    // Step 5: Hash the password for security
hashedPassword = hashPassword(password)
    // Step 6: Store user information in the database
newUser = {
        "username": username,
        "email": email,
        "password": hashedPassword,
        "registrationDate": currentDate()
    }
    saveUserToDatabase(newUser)
    // Step 7: Send confirmation email (optional)
sendConfirmationEmail(email)
    // Step 8: Display success message
    DISPLAY "Registration successful, please verify your email to complete registration"
```