Sri Lanka Institute of Information Technology

Information Technology Project (IT2080)

Batch ID - Y2.S2.WD.IT.10.01
Group ID - T141

Newton Electricals – Electrical Service Management System

Submitted by:

	Name with Initials	Registration Number	Contact Number	Email	Functionality
1.	Jayasinghe J.A.D.T.S	IT22028464	0761207101	it22028464@my.sliit.lk	User Management
2.	Wimalarathna B.P.K	IT22059604	0718427062	it22059604@my.sliit.lk	Career Management
3.	Gimsara W.T.G	IT22898098	0773479865	it22898098@my.sliit.lk	Feedback Management
4.	Galappaththi A.G.R.S	IT22345578	0710798661	it22345578@my.sliit.lk	Inventory Management
5.	P.R.Reid	IT22575944	0711201115	it22575944@my.sliit.lk	Service Schedule Management
6.	Dayarathne R.D.T.N	IT22578396	0775160412	it22578396@my.sliit.lk	Package Management
7.	Perera M.M.D	IT22581716	0760031024	it22581716@my.sliit.lk	Project Management
8.	Madhubhashana M D H P	IT22322876	0766803068	it22322876@my.sliit.lk	Order Management

▪ User Stories

Name	User Stories	Branch Name
User Management IT22028464	"As a registered user, I want to log into my account,So that I can access my Profile." "As a user, I want to be able to create an account easily so that I can access all features that are available." "As a user manager, I want to log into system as admin, So that I can manage users and their permissions." "As a registered user, I want to delete my account, So that I can delete my account whenever I want"	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical
Career Management IT22059604	"As a registered user, I want to create a comprehensive profile, so my skills and experiences are accurately represented." "As an HR manager, I want login to the System, So that Add the current job opportunities to the page " "As an employee, I want to view current job opportunities, so I can explore potential career advancements within Newton Electricals." "As an HR manager, I want to post job openings, so they are visible to all employees." "As an employee, I want to search for jobs based on specific criteria (e.g., department, location), so I can find relevant opportunities efficiently." "As a client, I want to access employee	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

	resumes, so I can evaluate potential candidates for electrical services." "As a HR manager, I want to communicate with clients and applicants seamlessly through the system, so I can schedule interviews and discuss job opportunities" "As an employee, I want to receive notifications on important events (e.g., new job applications, service request updates), so I can stay informed and responsive. "As an HR manager, I want to track task completion within the system, so I can assess employee performance and generate reports."	
Feedback Management IT22898098	As a user, I want to submit feedback easily: ● Description: Users should have a simple and intuitive way to provide feedback, whether it's through a form on the website, an in-app feedback option, or via email. ● Value: This allows users to share their thoughts and suggestions without encountering barriers, fostering a culture of open communication and engagement. As a customer, I want to receive notifications about my feedback: ● Description: Customers should receive notifications confirming receipt of their feedback and providing updates on its status, such as when it's being reviewed or resolved. ● Value: This keeps customers informed and reassured that their feedback is being taken seriously, enhancing their trust and satisfaction with the company. As an administrator, I want to prioritize	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

	feedback submissions: ● Description: Administrators should be able to review feedback submissions and prioritize them based on factors like urgency, impact, or frequency. ● Value: This allows administrators to focus on addressing the most critical feedback first, improving the efficiency and effectiveness of their response process. As a user, I want to filter feedback by different criteria: ● Description: Users should have the ability to filter feedback based on various criteria such as date, category, sentiment, or status. ● Value: This helps users quickly find the feedback that is most relevant to them, enabling them to focus on specific areas of interest or concern. As a administrator, I want to generate reports on feedback trends: ● Description: administrator should have access to reports and analytics that highlight trends and patterns in feedback submissions over time. ● Value: This provides valuable insights into customer preferences, pain points, and satisfaction levels, enabling informed decision-making and strategic planning.	
Inventory Management IT22345578	As an Inventory Manager,I want to easily add new items to the inventory, specifying details such as item name,description, quantity, unit of measurement, and supplier information,So that I can keep track of the available stock. As an Inventory Manager,I want to be able	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

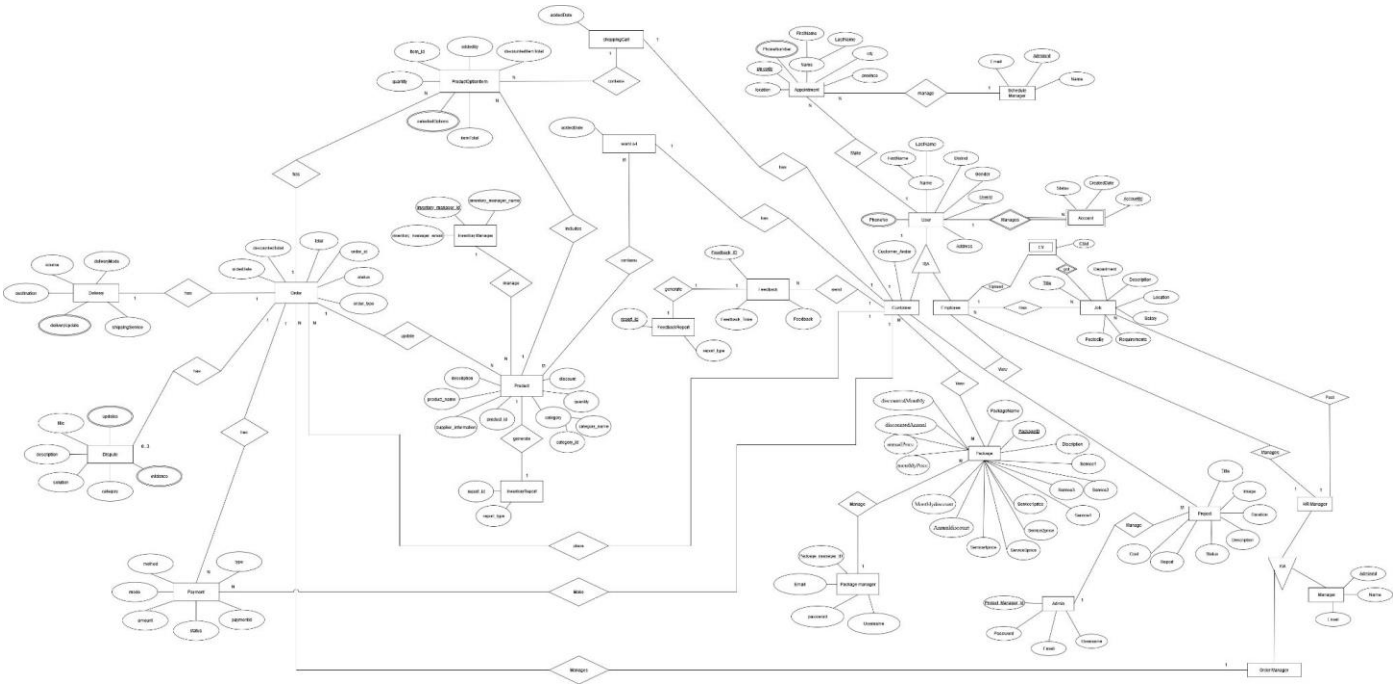
	to categorize inventory items based on their type and usage,So that I can easily locate and manage different types of items. As an Inventory Manager,I want the ability to update the quantity of items in real-time as new stock is received or items are used in service requests or maintenance activities, So that I can identify any discrepancies or issues in the system. As an Inventory Manager,I want to generate detailed reports on inventory levels, usage trends, and order history to aid in decision-making and strategic planning, So that I can review the usage patterns and plan for restocking as needed.	
Service Schedule Management IT22575944	“As a user, I want to access a user-friendly appointment booking interface so that I can book my appointments easily.” “As a user, I want to easily reschedule appointments with minimal hassle so that I can manage my schedule effectively.” “As a user, I want to have a straightforward process for canceling appointments so that I can adjust my schedule as needed without any complications.” “As a user, I want to access my booking details easily so that I can review and stay informed about my appointments.” “As a schedule manager, I want the capability to approve or reject bookings so that I can ensure that all appointments align with our resources, policies, and	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

	availability." "As a schedule manager, I want to be able to search for bookings so that I can quickly locate specific appointments, manage conflicts, and provide efficient assistance to users or clients."	
Package Management IT22578396	"As a Package Manager, I want to be able to add new package to the system,So that I can attract more Customers " "As a package manager, I want to be able to edit the details of a package in the system, So that I can update its information as needed." "As a Package Manager, I want to be able to delete the packages from the system, So that I can remove packages according to the season" "As a Customer, I want the ability to easily access and view the list of all packages, So that I can select a suitable package" "As a Customer, I want the ability to easily access and view the details of all packages, So that I can learn more about its features and benefits " "As a Customer, I want the ability to easily select subscription plans monthly or annually, So that I can choose the plan that best fits my needs and budget. " "As a package manager, I want to be able to generate the report of newly added packages, deleted packages, updated packages, So that I can effectively communicate changes to users, maintainers, and stakeholders."	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

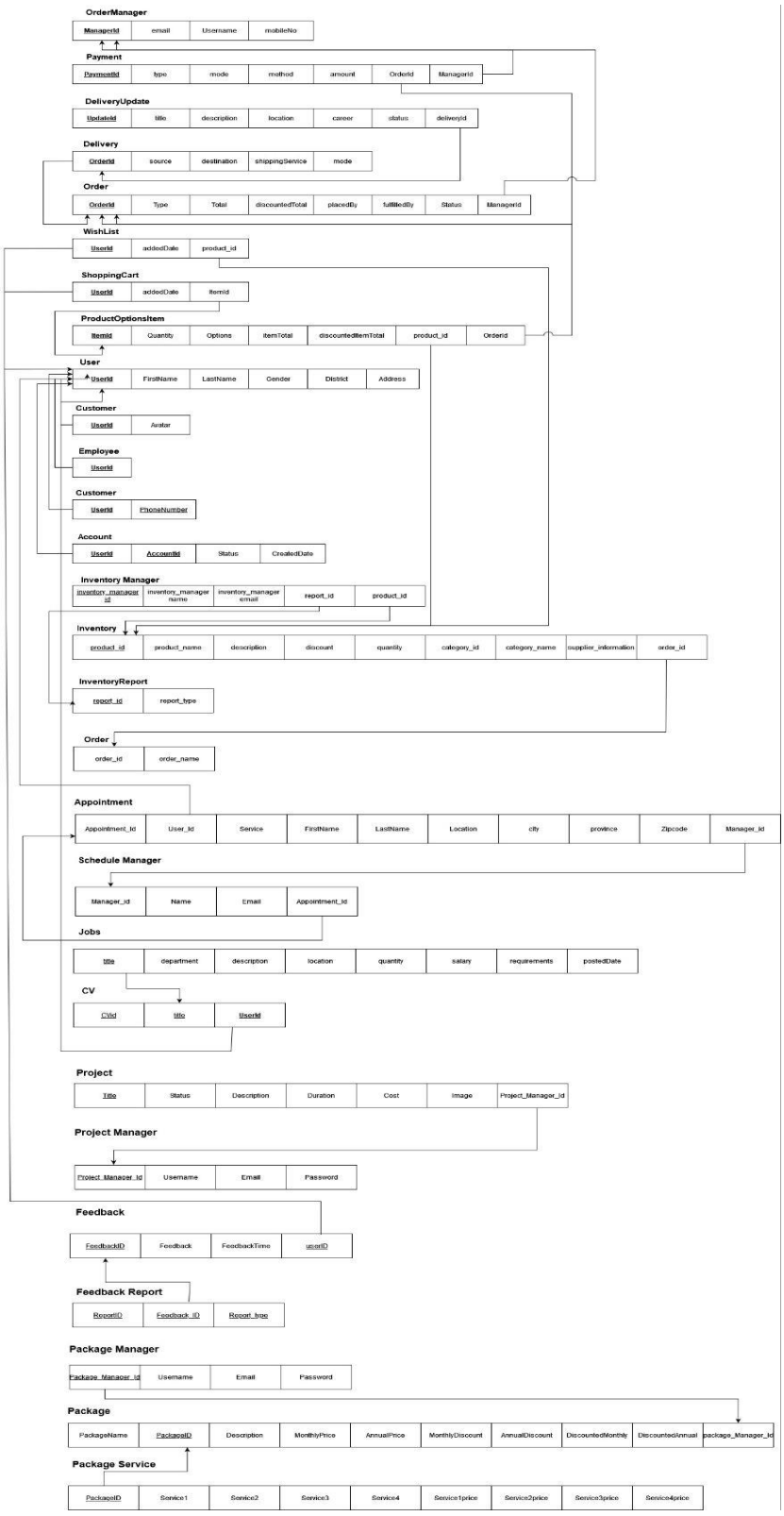
Project Management IT22581716	"As a Project Manager, I need to log in to the system, So that I can handle all project-related tasks effectively." "As a Project Manager, I want to reset my password, So that I can reset my password if I forget it." "As a Project Manager, I want to expected details of completed and in-progress projects are to be entered into the system, So, that I can the customers as well as the employees of the company can give an idea about the projects of the company." "As a Project Manager, I want to view projects of the company, So I can get an idea about the current status of projects." "As a Project Manager, I want to update details of the projects, So after completing the project can be added to the relevant section." "As a Project Manager, I want to delete projects, So I can maintain the system very well." " As a Project Manager, I want to generate comprehensive reports on project status, resource utilization and budget allocation, So that I can assess project performance, identify potential bottlenecks and make informed decisions to optimize project outcomes." "As an Employee, I need to log in to the system, So that I can check the completed projects in the system to being able to solve the problems we have about ongoing projects."	https://github.com/ThisaraJayas/ITP_Project__NewtonElectrical

	“As an Employee, I want to reset my password, So that I can reset my password if I forget it.” “As an Employee, I want to be able to view assigned tasks and deadlines within ongoing projects, So that I can prioritize my work and ensure timely completion of projects deliverables.” “As a Customer, I need to log in to the system, So that I can check the projects in the system to get ideas for my awareness." “As a Customer, I want to reset my password, So that I can reset my password if I forget it.” “As a Customer, I want to be able to submit project requests or inquiries regarding electrical installations or upgrades, So that I can communicate my requirements and expectations to the project management team.” “As a Customer, I want to receive updates and notifications on the progress of my project, including milestones achieved and upcoming tasks, So that I can stay informed and track the status of my project without needing to follow up constantly.” “As a Project Manager, I want to log out from the system, So that it adds more security to the system.” “As an Employee, I want to log out from the system, So that it adds more security to the system.” “As a Customer, I want to log out from the system, So that it adds more security to the system.”	

- **ER diagram**



- **Normalized schema**



▪ Test case design, Network design

[User Management] IT22028464 - Jayasinghe J.A.D.T.S

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
UN01	First Name: Thisara Last Name: Jayasinghe Email: sasmitha@gmail.com Phone Number: 0761203833 Password: 123Thisara Confirm Password: 123Thisara	User Registration Successful	User Successfully Registered	Pass
UN02	First Name: Kamla Last Name: Silva Email: Sasmithagmail.com Phone Number: 0761203833 Password: 123Kamal Confirm Password: 123Kamal	User Registration Successful	User Registration Failed (Invalid Email)	Fail
UN03	First Name: Nimal Last Name: Dias Email: nimal@gmail.com Phone Number: 0761203333 Password: 123Nimal Confirm Password: 123Nimal	User Registration Successful	User Successfully Registered	Pass

UN04	First Name: Harry Last Name: Olcot Email: harry@gmail.com Phone Number: 0721203533 Password: 123Harry Confirm Password: 123Harry	User Registration Successful	User Successfully Registered	Pass

[Career Management] IT22059604-Wimalarathna B.P.K

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
JL01	Title:Job1 Department: Finance Description: Financial Job Location: Colombo Salary: LKR50000 Requirements: Degrees Posted By: Nimali Posted Date: 2024-11-22 14:41:12	Job listing successfully added	Job listing successfully added.	Pass
JL02	Title: Job2 Department: IT Description: IT Job Location: Kandy Salary: LKR60000 Requirements: IT Degree Posted By: Saman Posted Date:	Job listing successfully added	Job listing successfully added	Pass

	2024-12-01 10:15:00			
JL03	Title:Job3 Department: Finance Description: Financial Job Location: Colombo Requirements: Degrees Posted By: Nimali Posted Date: 2024-11-22 14:41:12	Job listing successfully added	Failed to add job listing	Fail
JL04	Title: Job4 Department: Sales Description: Sales Job Location: Kandy Salary: LKR52000 Requirements: Sales Degree Posted By: Sunil Posted Date: 2024-12-15 11:45:00	Job listing successfully added	Job listing successfully added	Pass

[Feedback Management] IT22898098 - Gimsara W.T.G

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail

FB01	**First Name:** Saman **Last Name:** Perera **Email:** samanpr@gmail.com **Phone Number:** 0774125632 **Feedback Message:** The user interface is intuitive and easy to navigate. **Rating:** 5	Feedback Submission Successful.	Feedback Successfully Submitted.	Pass
FB02	**First Name:** Kamal **Last Name:** Priyankar **Email:** kamalpriy.com **Phone Number:** 0742513265 **Feedback Message:** I encountered a bug when trying to submit a form. **Rating:** 2	Feedback Submission Failed.	Invalid Email Format.	Fail
FB03	**First Name:** Vinuri **Last Name:** Yapa **Email:** vinury@gmail.com **Phone Number:** 0710256231 **Feedback Message:** The loading time of the application is too slow. **Rating:**3	Feedback Submission Successful.	Feedback Successfully Submitted.	Pass
FB04	**First Name:** Pavan **Last Name:** visal **Email:** pavanvis@gmail.com **Phone Number:** 0777452145	Feedback Submission Successful.	Feedback Successfully Submitted.	Pass

	Feedback Message: I suggest adding more features to improve the overall user experience. **Rating:** 4			

[Inventory Management] IT22345578 - Galappaththi A.G.R.S

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
PA01	Product Titel :Rechargable Desk Lamp Product Description :Unic Reachargable Desk Lamp Product Price-Rs.2,639 Product Image Product Quantity-13 Product Availability-In Stock	After filling out the add product form and clicking the submit button, the window will close and redirect to the product page. The system will be added new product and update the store item.	Display the added details of product in the specific table, and update the store.	pass
PA02	Product Titel: Air Conditioner Product Description :Singer Air Conditioner - Inverter 24000 BTU (SAS-24V-INV) Product Price-Rs.339,999 Product Image Product Quantity-130 Product Availability-In	After filling out the add product form and clicking the submit button, the window will close and redirect to the product page. The system will be added new	Display error message "value must be less than orv equl 100" and redirect adding product form.	fail

	Stock	product and update the store item.		
PA03	Product Titel: Air Conditioner Product Description :Singer Air Conditioner - Inverter 24000 BTU (SAS-24V-INV) Product Price-Rs.339,999 Product Image Product Quantity-99 Product Availability-In Stock	After filling out the add product form and clicking the submit button, the window will close and redirect to the product page. The system will be added new product and update the store item.	Display the added details of product in the specific table, and update the store.	pass
PA04	Product Titel: Rechargable Desk Lamp Product Description :Unic Reachargable Desk Lamp Product Price-Rs.2,639 Product Image Product Quantity-200 Product Availability-In Stock	After filling out the add product form and clicking the submit button, the window will close and redirect to the product page. The system will be added new product and update the store item.	Display error message "value must be less than orv equl 100" and redirect adding product form.	fail

[Service Schedule Management] IT22575944 - P.R.Reid

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
AB01	First Name: John Last Name: Doe Address: 123 Main Street City: Panadura Province: Kaluthara Zip Code: 12345 Date: 2024-05-01 Contact Number: 1234567890	"Appointment successfully booked!"	"Appointment successfully booked!"	Pass
AB02	First Name: Alice Last Name: Smith Address: 456 Elm Avenue City: Kalagedihena Province: Gampaha Zip Code: 67890 Date: 2024-04-30 Contact Number: 071345678	"Appointment successfully booked!"	"Failed to book the appointment. Please check your details and try again."	Fail
AB03	First Name: Emily Last Name: Johnson Address: 789 Oak Lane City: Makadura Province: Mathara Zip Code: 1234 Date: 2024-05-02 Contact Number: 0772463597	"Appointment successfully booked!"	"Failed to book the appointment. Please check your details and try again."	Fail
AB04	First Name: Sarah Last Name: Adams Address: 567 Malpara Street City: Irakkamam Province: Ampara Zip Code: 45678 Date: 2024-05-05	"Appointment successfully booked!"	"Appointment successfully booked!"	Pass

	Contact Number: 0711526423			
AB05	First Name: David Last Name: Wilson Address: 890 Malabe Road City: Malabe Province: Colombo Zip Code: 89012 Date: 2024-06-01 Contact Number: 0772516843	"Appointment successfully booked!"	"Appointment successfully booked!"	Pass

[Package Management] IT22578396 - Dayarathne R.D.T.N

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
PK01	packageName: Maintenance packageId: P001 Description: In this package, customers benefit from proactive preventive maintenance service1: Equipment cleaning service2: Equipment lubricating service1price : 1100 service2price : 900 Monthly price: 8000 Annual price : 24000 Monthly discount:5 Annual discount : 10 Discounted Monthly:7600 Discounted Annually :21600	“Package Added successfully”	“Package Added successfully”	Pass
PK02	packageName: Repairs packageId: P002 description:	“Package Added successfully”	"Please fill in all required fields."	Fail

	service2: breakdown repair service3: Equipment repair service4:Appliance Repair service1Price: 1000 service2Price: 700 service3Price: 200 service4Price: 400 monthly price: 8800 annual price: 26000 Annual discount:5% Monthly discount:10% discount monthly: 8000 discountedAnnually: 24000			
PK03	packageName: Rewiring packageId:P003 description: In this package, we offer specialized kitchen rewiring and bathroom rewiring service1:Kitchen Rewiring service2: Bathroom Rewiring service1Price:4000 service2Price: 5000 monthlyPrice: 34000 annualPrice: 100000 Annual discount:5% Monthly discount:10% discount monthly: 32300 discounted Annual:90000	"Package Added successfully"	"Package Added successfully"	Pass

PK04	packageName: Package four packageId:P004 description: This package grant customer four services with discounts. service1:First service service2: Second service service1Price:100 service2Price: 100 monthlyPrice: 800 annualPrice: 2400 Annual discount:5% Monthly discount:10% discount monthly: 760 discounted Annual:2280	"Package Added successfully"	"Package Added successfully"	Pass

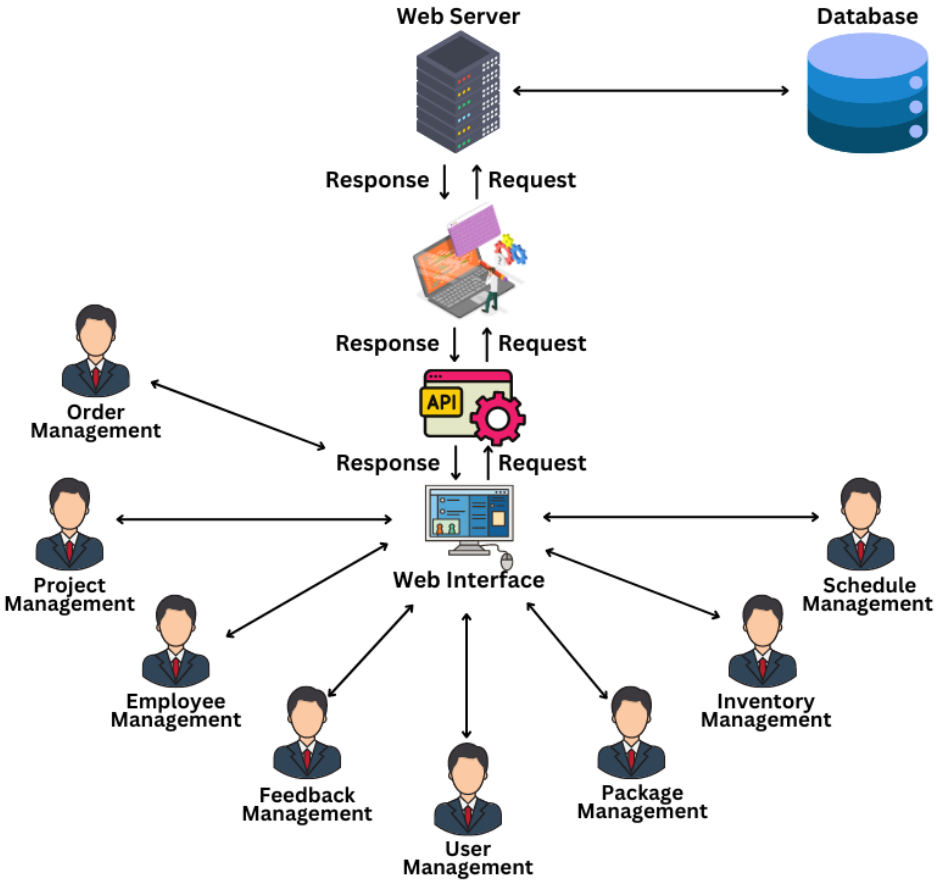
[Project Management] IT22581716 - Perera M.M.D

Test ID	Test Input	Expected Output	Actual Output	Result Pass/Fail
PM01	Title : CEB Substation - Kalavana Status : Ongoing Description : New Kalavana Sub Station LV wiring at Kukule. MR Construction (PVT) LTD Duration : 8 Months Cost : 85000000	Project Added Successfully	Project Details Successfully Added.	Pass

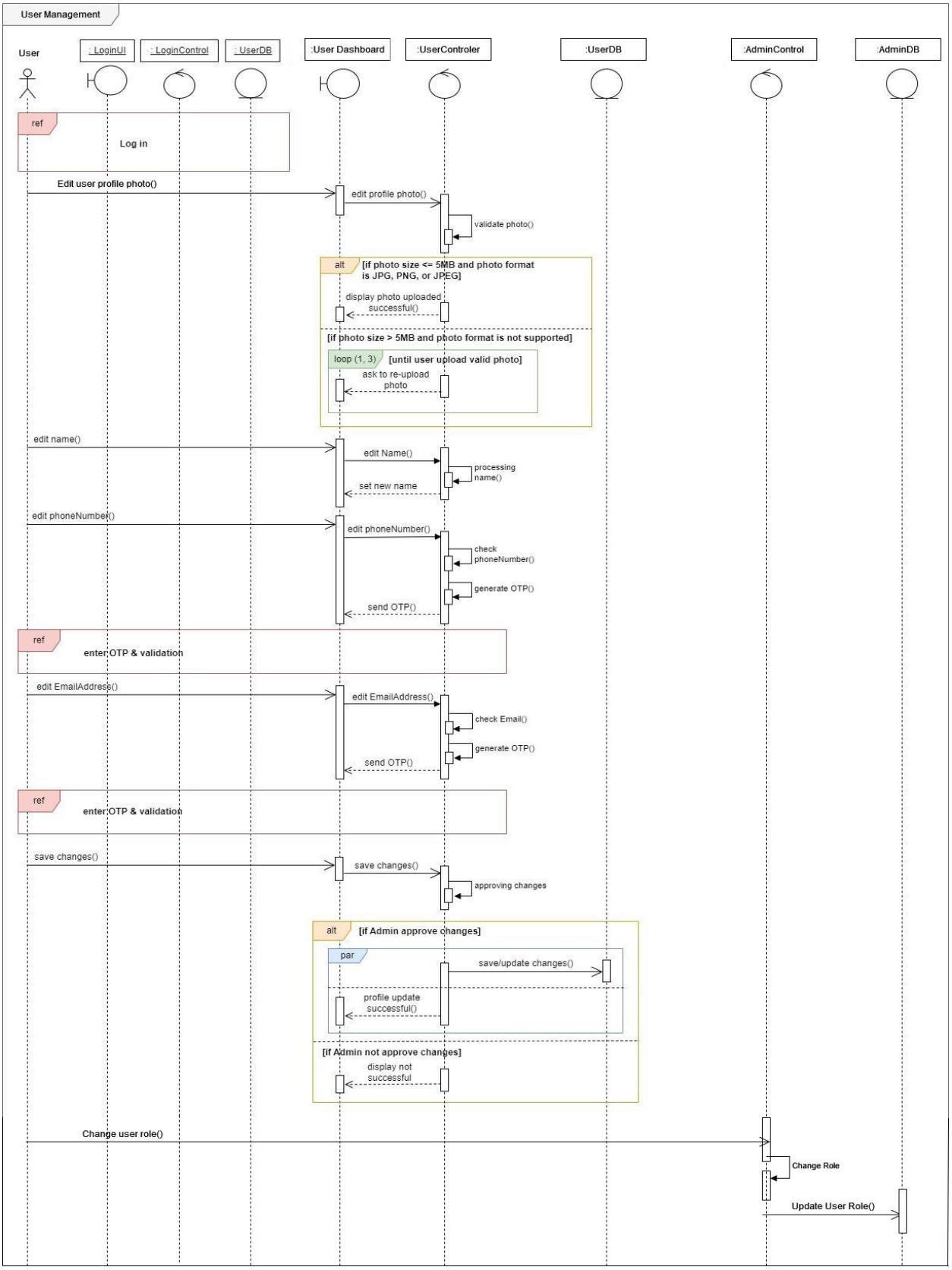
	Image : 1713316857276substation.jpg			
PM02	Title : Heritance Hotel - Ahungalla Status : Previous Description : Electrical Wiring Sub Contract under Wikkramage Engineering (PVT) LTD Duration : 2 Years Cost : 85000000 Image : 1713316484226heritance-ahungalla.jpg	Project Added Successfully	Project Details Successfully Added.	Pass
PM03	Title : Xavier's Church - Nuwara Eliya Status : Previous Description : 10+ Million worth rewiring of the Main Building of Xavier's Church. Panel boards newly installed. Duration : 1 Year Cost : 15000000 Image : 1713316244519LED-CHurch-Lighting-Design.jpg	Project Added Successfully	Project Details Successfully Added.	Pass

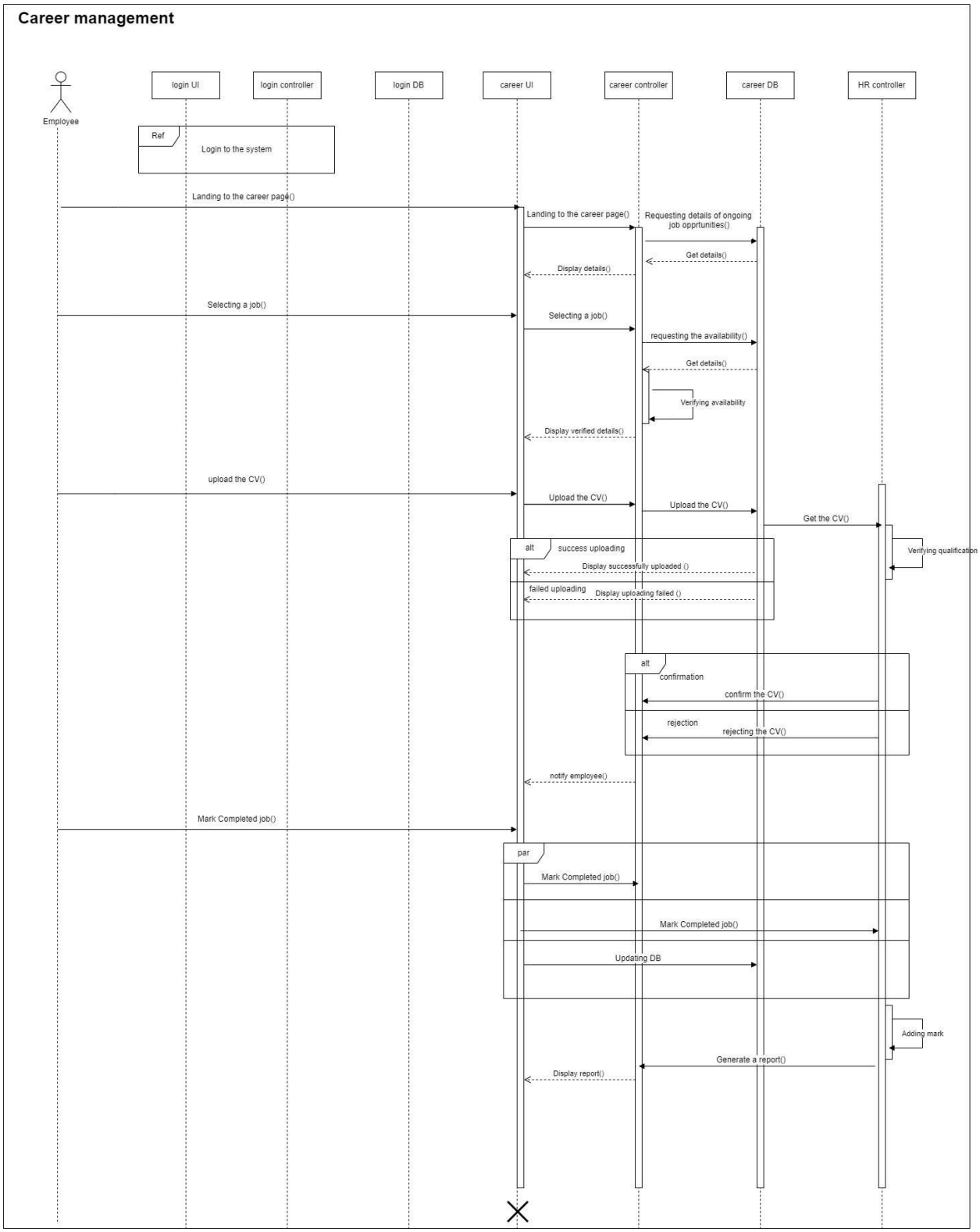
PM04	Title : Ceylon Villa - Badulla Status : Previous Description : Electrical Wiring Sub Contract under Rajawardhana Engineering (PVT) LTD Duration : 8 Months Cost : $55000000 Image : 1713316042034Amazing-cabins-3.jpg	Project Added Successfully	Failed to Add Project Details	Fail

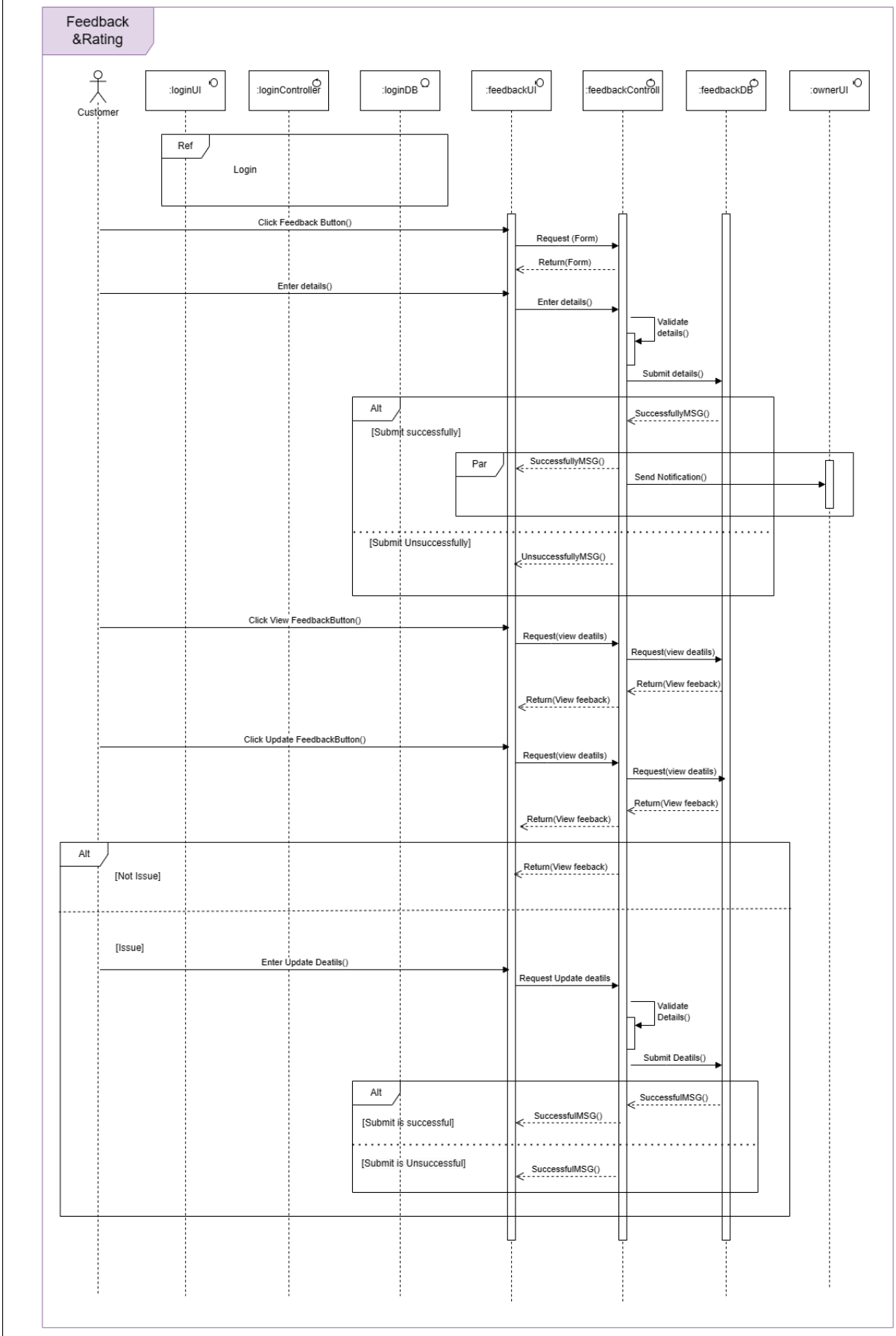
- **High-level system design diagram**

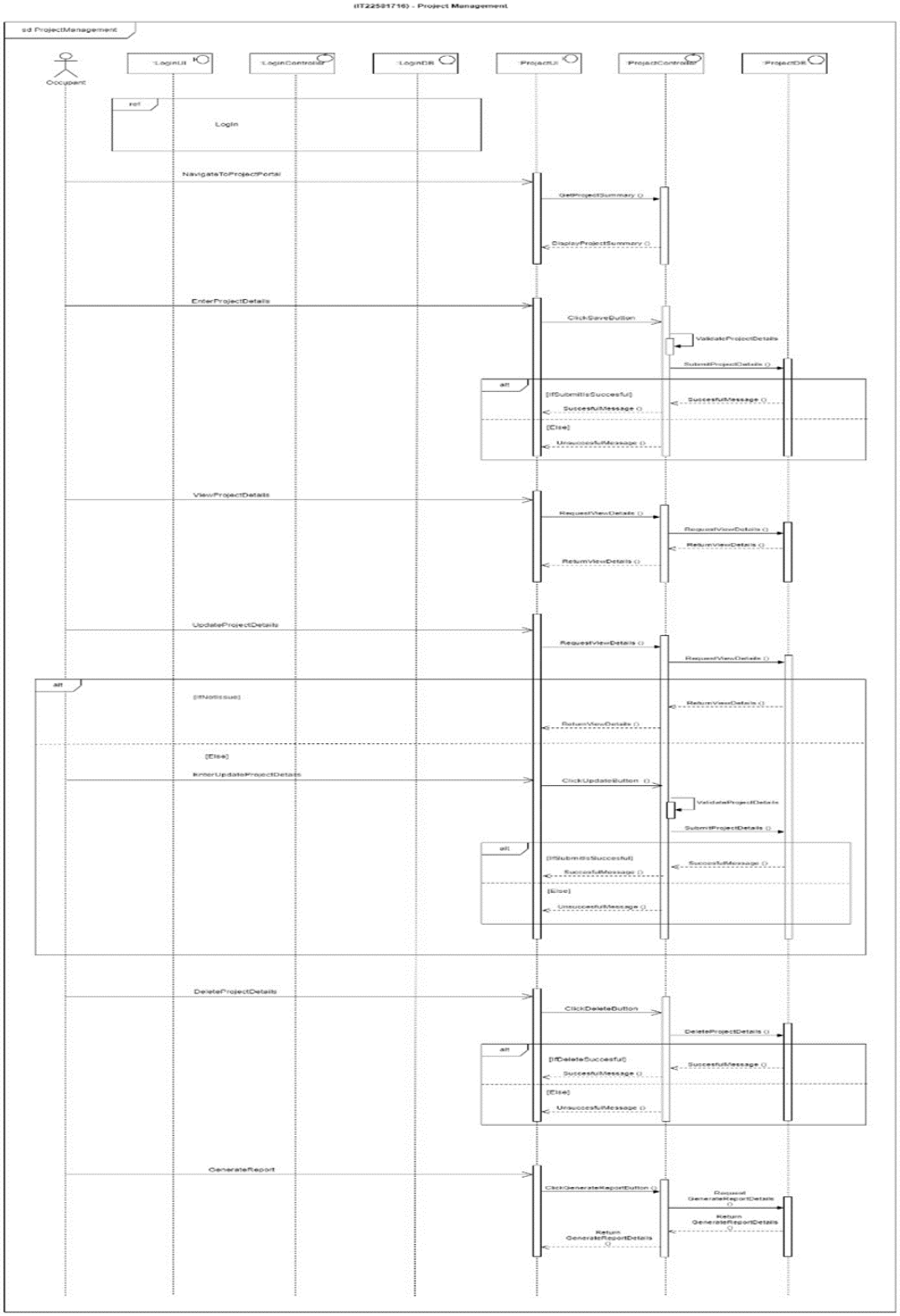


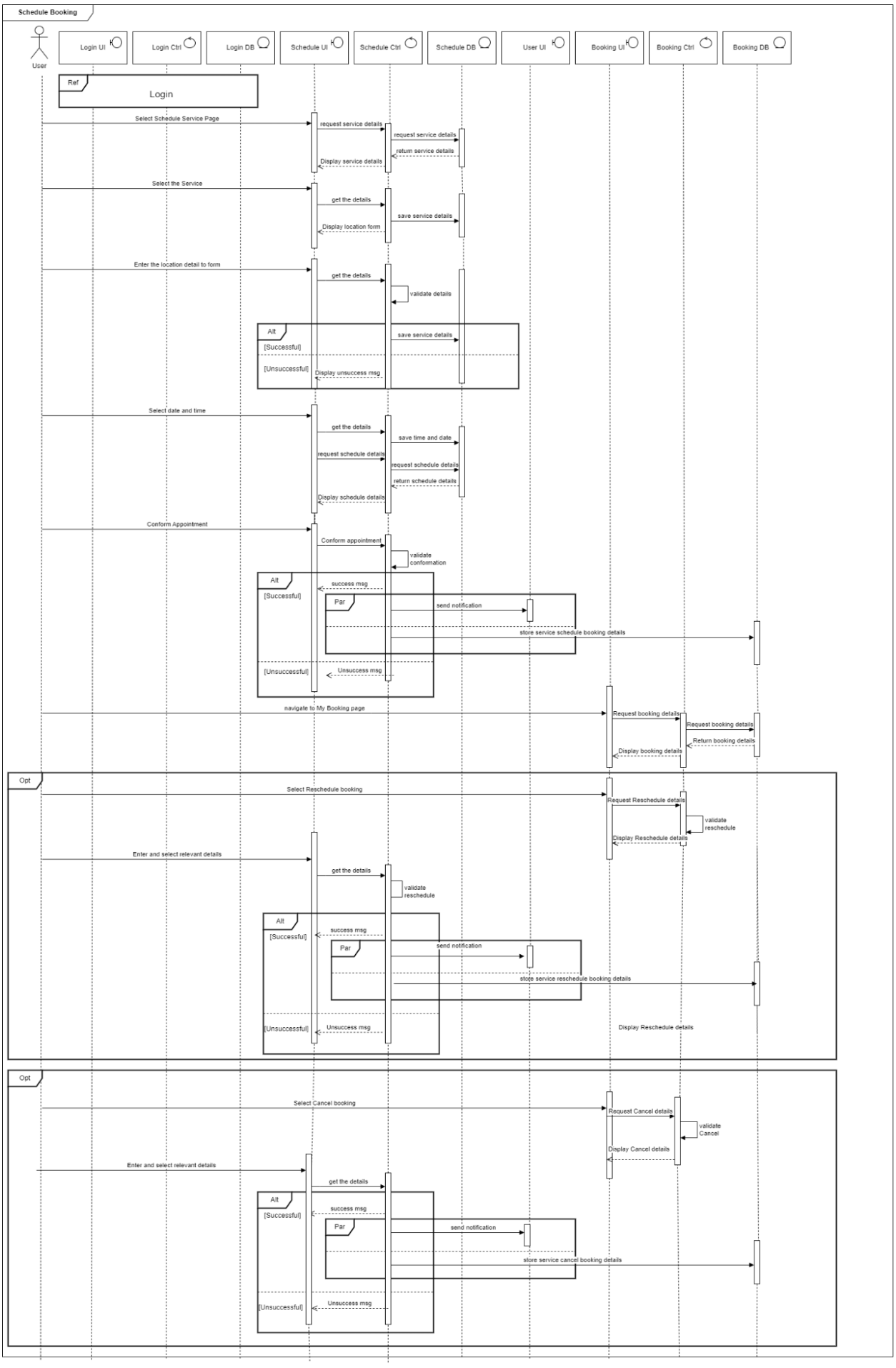
▪Sequence Diagram

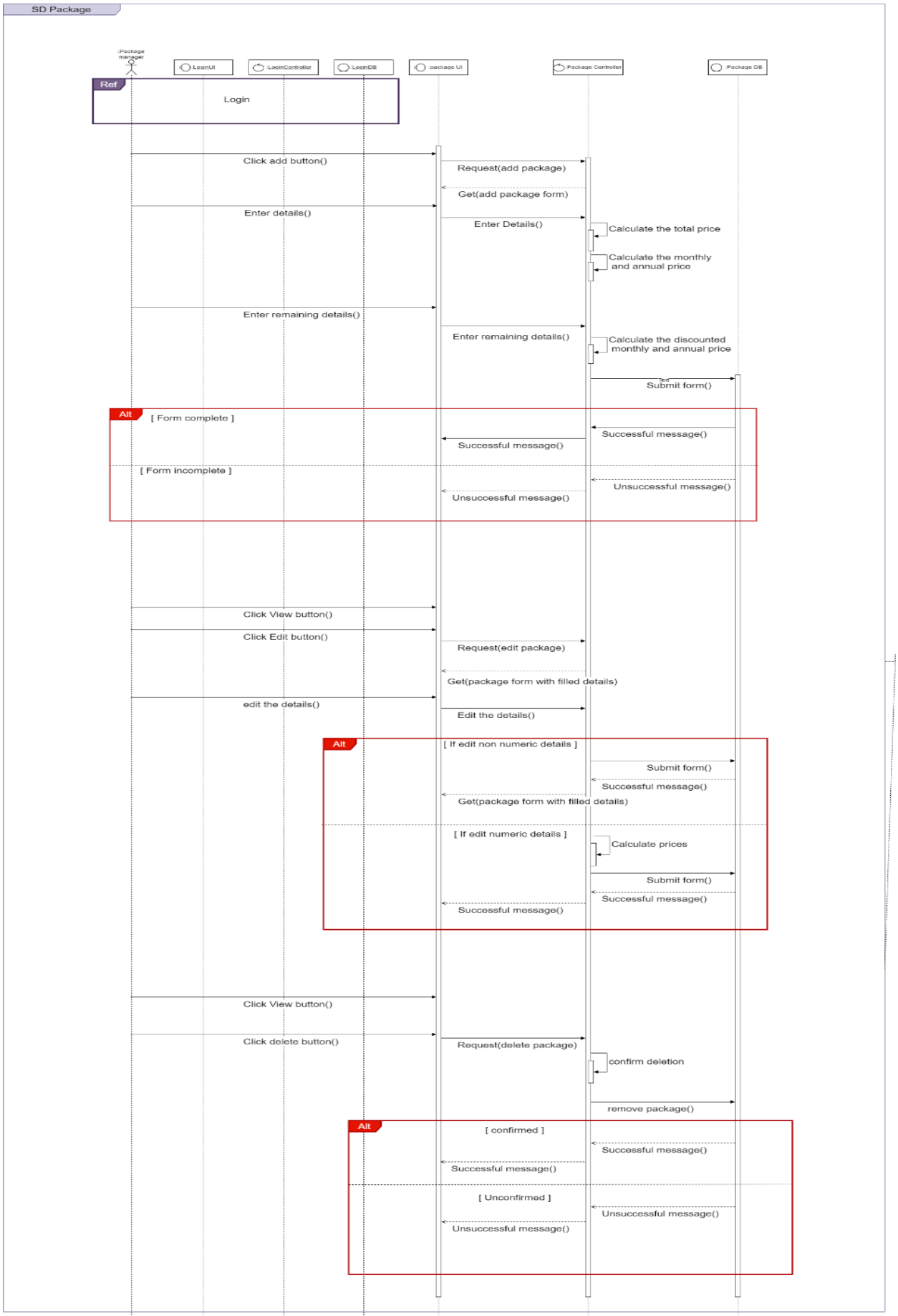



Career management
Employee
login UI
login controller
login DB
career UI
career controller
career DB
HR controller
Ref
Login to the system
Landing to the career page()
Landing to the career page()
Requesting details of ongoing job opprtunities()
Get details()
Display details()
Selecting a job()
Selecting a job()
requesting the availability()
Get details()
Verifying availability
Display verified details()
upload the CV()
Upload the CV()
Upload the CV()
Get the CV()
Verifying qualification
alt
success uploading
Display successfully uploaded ()
failed uploading
Display uploading failed ()
alt
confirmation
confirm the CV()
rejection
rejecting the CV()
notify employee()
Mark Completed job()
par
Mark Completed job()
Mark Completed job()
Updating DB
Adding mark
Generate a report()
Display report()


Feedback &Rating
Customer
:loginUI
:loginController
:loginDB
:feedbackUI
:feedbackControll
:feedbackDB
:ownerUI
Ref
Login
Click Feedback Button()
Request (Form)
Return(Form)
Enter details()
Enter details()
Validate details()
Submit details()
Alt
[Submit successfully]
SuccessfullyMSG()
Par
SuccessfullyMSG()
Send Notification()
[Submit Unsuccessfully]
UnsuccessfullyMSG()
Click View FeedbackButton()
Request(view deatils)
Request(view deatils)
Return(View feeback)
Return(View feeback)
Click Update FeedbackButton()
Request(view deatils)
Request(view deatils)
Return(View feeback)
Return(View feeback)
Alt
[Not Issue]
Return(View feeback)
[Issue]
Enter Update Deatils()
Request Update deatils
Validate Details()
Submit Deatils()
Alt
[Submit is successful]
SuccessfulMSG()
SuccessfulMSG()
[Submit is Unsuccessful]
SuccessfulMSG()


(IT22581716) - Project Management
sd ProjectManagement
Occupant
:LoginUI
:LoginController
:LoginDB
:ProjectUI
:ProjectController
:ProjectDB
ref
Login
NavigateToProjectForm()
GetProjectSummary()
DisplayProjectSummary()
EnterProjectDetails()
ClickSaveButton()
ValidateProjectDetails()
SubmitProjectDetails()
alt
[IfSubmitIsSuccessful]
SuccessfulMessage()
SuccessfulMessage()
[Else]
UnsuccessfulMessage()
ViewProjectDetails()
RequestViewDetails()
RequestViewDetails()
ReturnViewDetails()
ReturnViewDetails()
UpdateProjectDetails()
RequestViewDetails()
RequestViewDetails()
alt
[IfNotIssue]
ReturnViewDetails()
ReturnViewDetails()
[Else]
EnterUpdateProjectDetails()
ClickUpdateButton()
ValidateProjectDetails()
SubmitProjectDetails()
alt
[IfSubmitIsSuccessful]
SuccessfulMessage()
SuccessfulMessage()
[Else]
UnsuccessfulMessage()
DeleteProjectDetails()
ClickDeleteButton()
DeleteProjectDetails()
alt
[IfDeleteSuccessful]
SuccessfulMessage()
SuccessfulMessage()
[Else]
UnsuccessfulMessage()
GenerateReport()
ClickGenerateReportButton()
Request GenerateReportDetails()
Return GenerateReportDetails()
Return GenerateReportDetails()


Schedule Booking
User
Login UI
Login Ctrl
Login DB
Schedule UI
Schedule Ctrl
Schedule DB
User UI
Booking UI
Booking Ctrl
Booking DB
Ref
Login
Select Schedule Service Page
request service details
request service details
return service details
Display service details
Select the Service
get the details
save service details
Display location form
Enter the location detail to form
get the details
validate details
Alt
[Successful]
save service details
[Unsuccessful]
Display unsuccess msg
Select date and time
get the details
save time and date
request schedule details
request schedule details
return schedule details
Display schedule details
Conform Appointment
Conform appointment
validate conformation
Alt
success msg
[Successful]
Par
send notification
store service schedule booking details
[Unsuccessful]
Unsuccess msg
navigate to My Booking page
Request booking details
Request booking details
Return booking details
Display booking details
Opt
Select Reschedule booking
Request Reschedule details
validate reschedule
Display Reschedule details
Enter and select relevant details
get the details
validate reschedule
Alt
success msg
[Successful]
Par
send notification
store service reschedule booking details
[Unsuccessful]
Unsuccess msg
Display Reschedule details
Opt
Select Cancel booking
Request Cancel details
validate Cancel
Display Cancel details
Enter and select relevant details
get the details
Alt
success msg
[Successful]
Par
send notification
store service cancel booking details
[Unsuccessful]
Unsuccess msg


SD Package
:Package manager
:LoginUI
:LoginController
:LoginDB
:package UI
:Package Controller
:Package DB
Ref
Login
Click add button()
Request(add package)
Get(add package form)
Enter details()
Enter Details()
Calculate the total price
Calculate the monthly and annual price
Enter remaining details()
Enter remaining details()
Calculate the discounted monthly and annual price
Submit form()
Alt
[Form complete]
Successful message()
Successful message()
[Form incomplete]
Unsuccessful message()
Unsuccessful message()
Click View button()
Click Edit button()
Request(edit package)
Get(package form with filled details)
edit the details()
Edit the details()
Alt
[If edit non numeric details]
Submit form()
Successful message()
Get(package form with filled details)
[If edit numeric details]
Calculate prices
Submit form()
Successful message()
Successful message()
Click View button()
Click delete button()
Request(delete package)
confirm deletion
remove package()
Alt
[confirmed]
Successful message()
Successful message()
[Unconfirmed]
Unsuccessful message()
Unsuccessful message()

▪ Innovative parts of the project

Reward Points functionality: Special reward functionality where customers will be able to redeem reward packages based on the amount of rewards they have collected while using the system, where rewards are calculated based on the system login frequency and service schedules.

Customizable User Profiles: Allowing users to customize their profiles with avatars, bios, and personal preferences adds a personal touch to the platform.

Automated email: Integrating a system for sending automated email to customers for every successful transaction

Graph Analysis: Integrating a graph analysis system to read financial data and generate charts for visualizing trends and patterns offers a comprehensive approach to understanding complex datasets, this system enables users to gain deep insights into financial dynamics, facilitating informed decision-making and strategic planning..

▪ commercialization

The process of bringing the designed system to market, making it accessible to clients, and guaranteeing its profitability is known as commercialization of the Electrical and Service Management System. The following is a synopsis of the commercialization plan:

Product Development: To make sure the system satisfies the stated specifications and intended goals, it must be thoroughly designed, tested, and refined before going into commercialization.
Strategy for Pricing: Decide on the system's pricing approach. This might include different pricing structures like tiers of pricing determined by features, one-time purchases, or subscription-based pricing.

Advertising and Promotion:
Target Audience: Determine who the target market is for electrical services, including both home and business clients.

Channels for Promotion: Use a range of marketing methods to advertise the system, including social media, email marketing, web advertising, and industry alliances.
Convey the advantages of the system, highlighting its capacity to increase customer happiness, efficiency, and communication.

Launch Event: To introduce the system to prospective clients and create excitement, plan a launch event.

Distributing and Selling:
Direct Sales: Via the sales staff or website of Newton Electrical, make direct sales of the system.
Reseller Partnerships: Collaborate with other companies or online stores to offer the system via their distribution networks.

Marketplace Online: To expand the system's reach and visibility, list it on marketplaces online.
Client Assistance:
Instruction: Give clients instruction on how to use the system efficiently.
Technical Support: Provide technical assistance to resolve problems and respond to queries from clients.
Feedback Mechanism: To get customer insights and keep the system improving, set up a feedback mechanism.
New Features: Based on user input and market developments, the system will be updated frequently and expanded with new features.
Investigate potential avenues for market or regional expansion.
collaborations: To improve the capabilities of the system, establish collaborations with other service providers or IT firms.

Observation and Assessment:
Metrics of Performance: Establish the meaning of key performance indicators (KPIs) such customer happiness, client lifetime value, and cost of acquisition.
Frequent Assessment: Assess the system's effectiveness on a regular basis using these measures, and adapt as needed to achieve better results.
Newton Electrical can effectively launch the Electrical and Service Management System on the market by adhering to this commercialization plan, which will guarantee client adoption and the system's continued success in the market.

1.User Management System Development Progress -

IT22028464 - Jayasinghe J.A.D.T.S

Outline

User Management within Newton Electricals' system handles user registration, authentication, and role-based access for unregistered users, registered users, employees, and user managers. It offers simplified account creation, secure authentication methods, and user-friendly dashboard navigation for admins. User managers oversee user data, assign roles, and generate reports, while employees manage their profiles. Additionally, customers will have their own user profiles and customer functionalities in the system. A unique reward point system incentivizes user engagement, enhancing the overall customer experience. This section emphasizes the development of the user management system, focusing on finished features, ongoing projects, SQL queries utilized, and algorithms implemented.

1.1 Features' Completion Level:

Finished Features:

- successfully finalized the user registration, login, and profile sections, catering to both customers and administrators. Moreover, integrated a password reset feature, ensuring users receive reset links promptly to their email addresses for effortless account recovery.
- On the admin side, completed the functionality for modifying user roles through the user dashboard, providing administrators with efficient control over user permissions and access.
- Furthermore, in the admin dashboard, included features such as user count metrics, pie and bar charts, ensuring insights for informed decision-making.

Features Not Yet Finished:

- Reward feature where customers receive rewards based on their web activity.

1.2 SQL Queries Used (With a example):

```
import mongoose from "mongoose"

const userSchema = new mongoose.Schema({
        userId: {
                type:Number,
        },
        firstName:{
                type:String,
                required: true,
        },
        lastName:{
                type:String,
                required:true,
        },
        email: {
                type: String,
                required: true,
                unique: true,
        },
        mobileNumber:{
                type:String,
                required:true,
                Unique:true,
        },
        address:{
                type:String,
                default: ""
        },
        gender:{
                type:String,
        },
        password:{
                type:String,
                Required:true,
        },
        userType:{
                type:String,
                enum: ['Customer', 'Admin'],
```

```
            default:"Customer”
        },
        district:{
            type:String,
            default:" ",
        },
        avatar: {
            type: String,
            default: "https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_960_720.png"
        }
},{timestamps: true})

const User = mongoose.model('User',userSchema)
export default User
```

- **Registering an new User**
  INSERT INTO users(firstName, lastName, email, phoneNumber, password)
  VALUES ('Kamal', Perera, 'kamal@gamil.com', '0752356333', ‘123Kamal’);

- **Find Users**
  SELECT * FROM users
  WHERE userId= '1001';

- **Update User**
  UPDATE users
  SET userName= 'Kamal'
  WHERE userId= 1003;

- **Delete an User from the database**
  DELETE FROM users
  WHERE userId= 1005;

1.3 Algorithms, Flowcharts, Pseudocode:

1.3.1 Algorithms

1. Start
2. Request user action: login or register
3. If user chooses login:
   a. Input login credentials (email and password)
   b. Validate credentials against stored data
   c. If credentials are valid:
      - Check user role:
      - If user role is administrator, grant access to admin dashboard
      - If user role is customer, grant access to home page
   d. If credentials are invalid:
      - Display "Invalid username and password"
      - Limit login attempts to 3 times
      - If login attempts exceed limit, navigate to registration page
   e. If user clicks password reset, proceed to password reset page
4. If user chooses register:
   a. Prompt for required user information (first name, email, password, etc.)
   b. Validate input data (unique email, mobile number, etc.)
   c. If user data is valid, insert data into the database and grant access to home page or user dashboard
   d. If user data is invalid, provide feedback and allow user to retry registration
5. If logged-in user chooses to update profile:
   a. Prompt for user to enter new information (e.g., name, email, etc.)
   b. Validate updated user information and update data in the database
6. If logged-in user chooses to delete account:
   a. Prompt for confirmation
   b. If confirmed, delete user data from the database
   c. Provide confirmation message and navigate to login page
7. End

1.3.2 Pseudocode

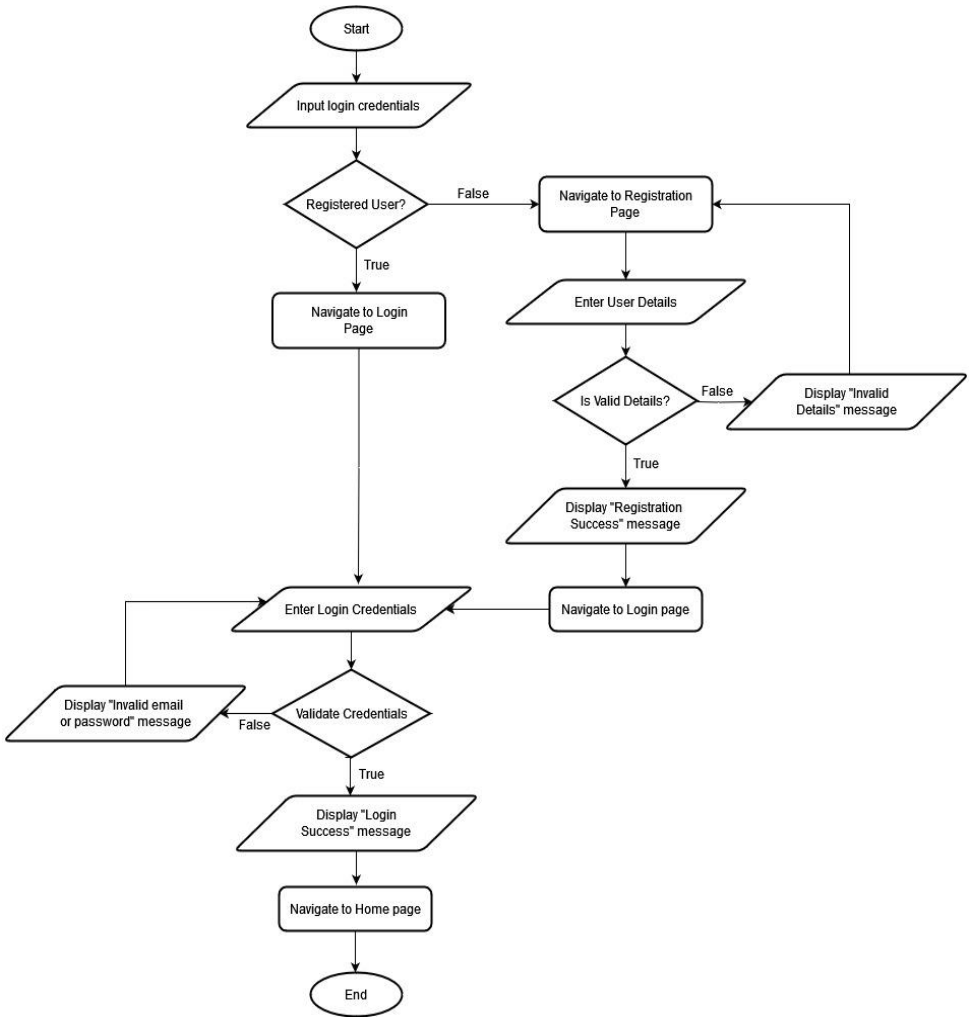
```
BEGIN
REQUEST user action: login or register
IF user action is login THEN
        INPUT login credentials (email and password)
        VALIDATE credentials against stored data
        IF credentials are valid THEN
        CHECK user role
        IF user role is administrator THEN
        GRANT access to admin dashboard
        ELSE IF user role is customer THEN
        GRANT access to home page
        END IF
        ELSE
        DISPLAY "Invalid username and password"
        LIMIT login attempts to 3 times
        IF login attempts exceed limit THEN
        NAVIGATE to registration page
        END IF
        END IF
        IF user clicks password reset THEN
        PROCEED to password reset page
        END IF
ELSE IF user action is register THEN
        PROMPT for required user information (first name, email, password, etc.)
        VALIDATE input data (unique email, mobile number, etc.)
        IF user data is valid THEN
        INSERT data into the database
        GRANT access to home page or user dashboard
        ELSE
        PROVIDE feedback and ALLOW user to retry registration
        END IF
END IF
IF logged-in user CHOOSES to update profile THEN
        PROMPT for user to enter new information (e.g., name, email, etc.)
        VALIDATE updated user information
        UPDATE data in the database
END IF
IF logged-in user CHOOSES to delete account THEN
```

PROMPT for confirmation
IF confirmed THEN
DELETE user data from the database
PROVIDE confirmation message
NAVIGATE to login page
END IF
END IF
END

1.4 Final Completion To-Do List:

- Reward feature where customers receive rewards based on their web activity.
- Implement the password encryption which hash the password in the database

Flow Chart



2.Career Management System Development Progress

IT22059604-Wimalarathna B.P.K

Outline

For businesses looking to optimize personnel development, retention, and acquisition, effective career management systems are vital resources. To improve customer satisfaction and operational efficiency, Newton Electricals understands the need of putting in place a strong career management system. With an emphasis on finished features, ongoing projects, SQL queries utilized, and algorithms implemented, this paper describes the development of the career management system.

2.1 Features' Completion Level:

Finished Features:

- UI and backend operations for the job listing and search functionality are finished.
- Task completion and reporting: Both the front-end and back-end activities have been finished.

Partially Finished Features:

- Recruiting Resumes: Backend processes finished, user interface pending.

Features Not Yet Finished:

- Communication Tools: UI and backend operations are still being worked on.
- Notifications: There are pending UI and backend actions.

2.2 Tasks Unfinished and Still Needing to Be Done:

- Tools of Communication:
- Full front-end and back-end functionality for email and phone integration.

- Notifications: Set up effective alerting mechanisms for new job applications, service request changes, and other relevant data.

2.3 SQL Queries Used (With a example):

```
// models/JobModel.js

import mongoose from 'mongoose';

const jobSchema = new mongoose.Schema({

    title: { type: String, required: true },
    department: { type: String, required: true },
    description: { type: String, required: true },
    location:{ type: String },
    salary:{ type: String},
    requirements:{ type: String},
    postedBy:{ type: String, required: true },
    postedDate:{ type: Date, default:Date.now },

    // Add other fields as needed for job postings

});

const Job = mongoose.model('Job', jobSchema);

export default Job;
```

- **Inserting a Job:**

```
INSERT INTO JobListings (Title, Department, Description, Location, Salary, Requirements, PostedBy, PostedDate)
VALUES ('Job1', 'Finance', 'Financial Job', 'Colombo', 'LKR50000', 'Degrees', 'Nimali', '2024-11-22 14:41:12');
```

- **Finding Job:**

```
SELECT * FROM JobListings WHERE Department = 'Finance';
```

- **Updating a Job(e.g., changing salary):**

```
UPDATE JobListings SET Salary = 'LKR60000' WHERE Title = 'Job1';
```

- **Deleting a Job:**

```
DELETE FROM JobListings WHERE Title = 'Job1';
```

2.4 Algorithms, Flowcharts, Pseudocode:

2.4.1 Algorithm

```
Function displayJobListings():
   jobList = retrieveJobListings()  // Retrieve job listings from the database
   if jobList is not empty:

      for each job in jobList:
         displayJob(job)  // Display each job listing
   else:
      displayMessage("No job listings found.")  // Display a message if no jobs are found

Function searchJobListings(criteria):
   jobList = retrieveJobListingsByCriteria(criteria)  // Retrieve job listings based on search
criteria
   if jobList is not empty:
      for each job in jobList:
         displayJob(job)  // Display each job listing that matches the criteria
   else:
      displayMessage("No job listings found matching the criteria.")  // Display a message if no
matching jobs are found

Function displayJob(job):
   display(job.title)
   display(job.department)
   display(job.description)
   display(job.location)
   display(job.salary)
   display(job.requirements)
   display(job.postedBy)
   display(job.postedDate)

Function retrieveJobListings():
   jobList = queryDatabase("SELECT * FROM JobListings")  // Query the database to retrieve
all job listings
   return jobList

Function retrieveJobListingsByCriteria(criteria):
   jobList = queryDatabase("SELECT * FROM JobListings WHERE " + criteria)  // Query the
database to retrieve job listings based on criteria
   return jobList
```

```
Function queryDatabase(sqlQuery):
    // Execute the SQL query on the database
    result = executeSQLQuery(sqlQuery)
    return result
```

2.4.2 Pseudocode

```
Start
Call searchJobListings(criteria)
End

Function searchJobListings(criteria)
    jobList = retrieveJobListingsByCriteria(criteria)
    if jobList is empty then
        displayMessage("No job listings found matching the criteria.")
    else
        Loop through each job in jobList
            displayJob(job)
        End Loop
    End if
End Function

Function displayJobListings()
    jobList = retrieveJobListings()
    if jobList is empty then
        displayMessage("No job listings found.")
    else
        Loop through each job in jobList
            displayJob(job)
        End Loop
    End if
End Function

Function retrieveJobListings()
    // Function body for retrieving job listings goes here
End Function

Function retrieveJobListingsByCriteria(criteria)
```

```
    // Function body for retrieving job listings by criteria goes here
End Function

Function displayJob(job)
    // Function body for displaying job details goes here
End Function

Function displayMessage(message)
    // Function body for displaying a message goes here
End Function
```

2.5 Final Completion To-Do List:

- Full user interface and backend for resume retrieval.
- Use communication tools, such as integrated email and phone functions.
- Provide a reliable notification mechanism to encourage user participation and responsiveness.

3.Feedback Management System Development Progress

IT22898098 - Gimsara W.T.G

Outline

Newton Electricals' Feedback Management System development progresses steadily, focusing on special solutions for efficient feedback handling within its service management system. Key features include database schema establishment and user-friendly interface creation for seamless feedback submission and management. Prioritizing user experience enhancements like email notifications and analytics tools, the Feedback Management System aims to optimize service quality and operational efficiency iteratively and to enhance customer satisfaction.

3.1 Features' Completion Level:

Finished Features:

- Feedback Submission Form.
- Implemented user feedback edit and deletion in user side.
- Completed feedback deletion in admin dashboard.
- Implemented feedback rating functionality.
- Additionally, implemented, pie chart, and report generation in the admin dashboard.
- Implemented full CRUD.
- Implemented a validation parts.
- Implement an email notification system to alert administrators about new feedback submissions promptly.

Features Not Yet Finished:

- Implement search functionality

3.2 SQL Queries Used (With a example):

```
import mongoose from "mongoose";

// Define the schema for feedback submissions
const feedbackSchema = new mongoose.Schema({
   firstName: {
      type: String,
```

```
        required: true
    },
    lastName: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    phoneNumber: {
        type: String,
        required: true
    },
    feedbackMessage: {
        type: String,
        required: true
    },
    rating: {
        type: Number,
        required: true,
        min: 1,
        max: 5
    },
    timestamp: {
        type: Date,
        default: Date.now
    }
});

// Create a Mongoose model for feedback submissions
const Feedback = mongoose.model('Feedback', feedbackSchema);

// Insert a new feedback submission
const newFeedback = new Feedback({
    firstName: 'Namal',
    lastName: 'Rathnayaka',
    email: 'namalrat@gmail.com',
    phoneNumber: '0774512563',
    feedbackMessage: 'This is a great feature!',
```

```
  rating: 5
});

newFeedback.save()
  .then(feedback => {
    console.log('Feedback submitted successfully:', feedback);
  })
  .catch(error => {
    console.error('Error submitting feedback:', error);
  });
```

- CREATE: Insert a new feedback entry into the database

```
INSERT INTO feedback (first_name, last_name, email, phone_number, feedback_message,
rating)
VALUES ('Namal', 'Rathnayaka', 'namalra@gmail.com', '0778562456', 'Great service, very
helpful staff!', 5);
```

- READ: Retrieve feedback entries from the database

```
SELECT * FROM feedback;
```

- UPDATE: Update an existing feedback entry in the database

```
UPDATE feedback
SET rating = 4, feedback_message = 'Good service, but room for improvement'
WHERE feedback_id = 1;
```

- DELETE: Delete a feedback entry from the database

```
DELETE FROM feedback
WHERE feedback_id = 2;
```

3.3 Algorithms, Flowcharts, Pseudocode:

3.3.1 Algorithms

1. Start: The beginning of the feedback management algorithm.
2. Request Feedback Action: Prompt the user to provide feedback.
3. Receive Feedback:
   a. User inputs feedback details including first name, last name, email, phone number, feedback message, and rating.
   b. Validate the input data for completeness and correctness.
   c. If the data is valid:
      - Insert the feedback data into the database.
      -Provide acknowledgment of successful feedback submission.

   d. If the data is invalid:
      - Display an error message indicating the required fields or format of data.
      -Allow the user to retry providing feedback.

4. View Feedback:
   a. Authorized users can access the feedback data.
   b. Retrieve feedback entries from the database.
   c. Display feedback entries to authorized users for review and analysis.

5. Update Feedback:
   a. Authorized users can update feedback entries if needed.
   b. Prompt the user to select the feedback entry to update.
   c. Allow the user to modify feedback details such as the feedback message or rating.
   d. Validate the updated data and update the corresponding entry in the database.

6. Delete Feedback:
   a. Authorized users can delete feedback entries.
   b. Prompt the user to select the feedback entry to delete.
   c. Confirm the deletion action.
   d. If confirmed, delete the selected feedback entry from the database.

7. End: The conclusion of the feedback management algorithm.

3.3.2 Pseudocode

```
BEGIN Feedback Management

  FUNCTION RequestFeedbackAction()
    DISPLAY "Please provide your feedback:"
    PROMPT "Enter your feedback message:"
    PROMPT "Rate your experience (1-5):"
    GET user input for feedback message and rating
    RETURN user input
  END FUNCTION

  FUNCTION ValidateFeedbackData(input)
    IF input is not empty AND rating is between 1 and 5 THEN
      RETURN true
    ELSE
      DISPLAY "Error: Please provide a feedback message and rating between 1 and 5."
      RETURN false
    END IF
  END FUNCTION

  FUNCTION InsertFeedbackIntoDatabase(feedbackData)
    INSERT feedbackData INTO feedback_table
  END FUNCTION

  FUNCTION ViewFeedback()
    SELECT * FROM feedback_table
    DISPLAY feedback entries
  END FUNCTION

  FUNCTION Main()
    DO
      RequestFeedbackAction()
      IF ValidateFeedbackData(input) THEN
        InsertFeedbackIntoDatabase(input)
        DISPLAY "Thank you for your feedback! Your input has been recorded."
      END IF
      PROMPT "Do you want to view feedback entries? (Y/N):"
      GET user input
      IF user input is "Y" THEN
```

```
            ViewFeedback()
        END IF
        PROMPT "Do you want to provide more feedback? (Y/N):"
        GET user input
    WHILE user input is "Y"
END FUNCTION

Main()

END Feedback Management
```

3.4 Final Completion To-Do List:

- Implement security measures to protect sensitive feedback data and ensure compliance with privacy regulations, such as data encryption and access controls.
- Add a search input field to the user interface where users can enter the username they want to search for.

4.Inventory Management System Development Progress -

IT22345578 - Galappaththi A G R S

Outline

Inventory management in an electrical service management system involves tracking and organizing electrical components, tools, and equipment. Key functions include real-time inventory tracking to prevent stockouts and identifying surplus or obsolete items. The system also categorizes items for easier management, integrates with accounting and project management modules, and generates reports on inventory status and cost analysis. Overall, it streamlines operations, reduces costs, and enhances productivity in electrical service management.

4.1 Features' Completion Level:

Finished Features:

- Completed store page
- Implemented admin dashboard to add product
- Implemented full CRUD

Features Not Yet Finished:

- Implement search functionality
- Implement store page

4.2 SQL Queries Used (With a example):

```
import mongoose from "mongoose";

const inventoryItemSchema = new mongoose.Schema({
   itemName: {
      type: String,
      required: true,
   },
   description: {
      type: String,
      required: true,
   },
   category: {
      type: String,
```

```
    required: true,
  },
  quantity: {
    type: Number,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
  supplier: {
    type: String,
    required: true,
  },
  reorderLevel: {
    type: Number,
    required: true,
  },
  // Add any other relevant fields for inventory management
}, { timestamps: true });

const InventoryItem = mongoose.model('InventoryItem', inventoryItemSchema);
export default InventoryItem;
```

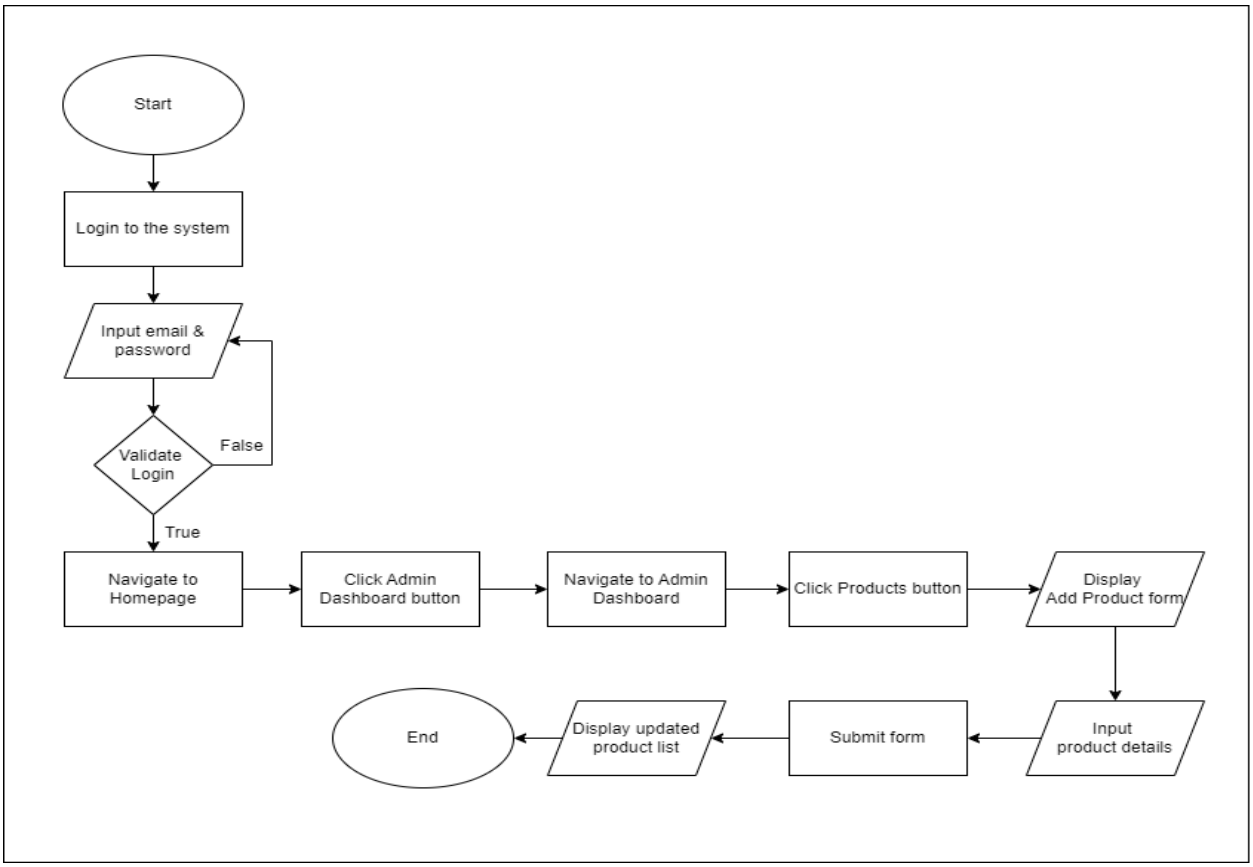
4.3 Algorithms, Flowcharts, Pseudocode:

4.3.1 Algorithms

1. Start
2. Login to the system
   a. Input login credentials (email and password)
   b. Validate credentials
   c. If credentials are valid:
      - Navigate to home page
   d. If credentials are invalid:
      - Display "Invalid username and password"
      - Navigate to login page
4. Click Admin Dashboard button
   a. Navigate to Admin Dashboard

5. Click Products button
   a. Display Add Product form
   b. Input product details
   c. Submit form
6. Display updated product list
7. End

4.3.2 Flowcharts



4.3.3 Pseudocode

BEGIN
Login to the system
INPUT login credentials (email and password)
IF Validate Login THEN
        NAVIGATE to Homepage
        CLICK Admin Dashboard button
        NAVIGATE to Admin Dashboard
        CLICK Products button
        DISPLAY Add Product form
    INPUT product details

```
        SUBMIT form
    DISPLAY  updated product list
ELSE
    DISPLAY "Invalid username and password"
        NAVIGATE to login page
END IF
END
```

4.4 Final Completion To-Do List:

- Address any outstanding punch list items, including minor repairs, adjustments, or touch-ups, to achieve project completion.
- Conduct a final inventory check to ensure all tools, equipment, and spare parts are accounted for and properly stored.
- Conduct a thorough inspection of all electrical components, installations, and systems to ensure they meet quality standards and comply with regulations.

5.Service Schedule Management System Development Progress -

IT22575944 - P.R.Reid

In the ongoing development of Newton Electrical's service schedule management system, users can easily book appointments by providing key details such as their name, address, date, and contact number. Upon successful booking, users receive immediate confirmation. They also have the flexibility to reschedule or cancel appointments as needed. Service schedule managers play a crucial role in the system by accepting and managing appointments efficiently, ensuring alignment with resources and policies. This system aims to streamline appointment processes, enhancing overall efficiency and customer satisfaction.

5.1 Features' Completion Level:

Finished Features:

- Appointment booking form and validation part have been implemented.
- My Booking page displaying booking details has been finished.
- Reschedule and cancelation options are now available.
- Appointment approval system is in place, showing status to users.
- Admin side features search and report functionality.

Features Not Yet Finished:

- Service Price Calculation.

5.3 SQL Queries Used (With a example):

```
import mongoose from "mongoose";

const appointmentSchema = new mongoose.Schema({
        select1: String,
        select2: String,
        description: String,
        firstName: String,
        lastName: String,
        address: String,
        city: String,
        province: String,
        zipcode: String,
        contactNum: String,
        ownProperty: Boolean,
        timeSlot: Date,
        status: {
        type: String,
```

```
        enum: ['pending', 'approved', 'canceled'],
        default: 'pending'
        }
});
const Appointment = mongoose.model('Appointment', appointmentSchema);
export default Appointment;
```

- **Insert an Appointment to the database**
  INSERT INTO appointments (select1, select2, description, firstName, lastName, address, city, province, zipcode, contactNum, ownProperty, timeSlot, status)

  VALUES ('repair wires', 'cooling', 'whole house', 'John', 'coner', '123 Main Street', 'panadura, 'Kaluthara', '12345', '0775426985', '1', '2024-04-23 10:00:00', 'pending');

- **Find appointments**
  SELECT * FROM appointments
  WHERE appointment_id = 'AP1254801562';

- **Update Appointment date in the database**
  UPDATE appointments
  SET timeSlot = '2024-05-01 09:00:00'
  WHERE appointment_id = 'AP1254801562';

- **Delete an appointment from the database**
  DELETE FROM appointments
  WHERE appointment_id = 'AP1254801562';

5.3 Algorithms, Flowcharts, Pseudocode:

5.3.1 Algorithms

Start: The beginning of the appointment management algorithm.

1. Request Appointment Action:
   - Prompt the user to schedule an appointment.

Schedule Appointment:

a. User fills in appointment details including first name, last name, address, city, province, zipcode, contact number, appointment date, and any additional details.

2. b. Validate the input data for completeness and correctness.
    - If the data is valid:
        - Save the appointment data into the database.
        - Provide acknowledgment of successful appointment scheduling.
    - If the data is invalid:
        - Display an error message indicating the required fields or format of data.
        - Allow the user to retry scheduling the appointment.

View Appointments:

a. Users can access their scheduled appointments.

b. Retrieve appointment entries from the database.

3. c. Display appointment details to users for review and management.

Reschedule Appointment:

a. User selects the appointment to reschedule.

b. Prompt the user to choose a new date and time for the appointment.

4. c. Validate the new date and time for availability and correctness.
    - If valid:
        - Update the appointment with the new date and time in the database.
        - Provide acknowledgment of successful rescheduling.
    - If invalid:
        - Display an error message indicating the unavailability or incorrectness of the chosen date and time.
        - Allow the user to retry selecting a new date and time.

Cancel Appointment:

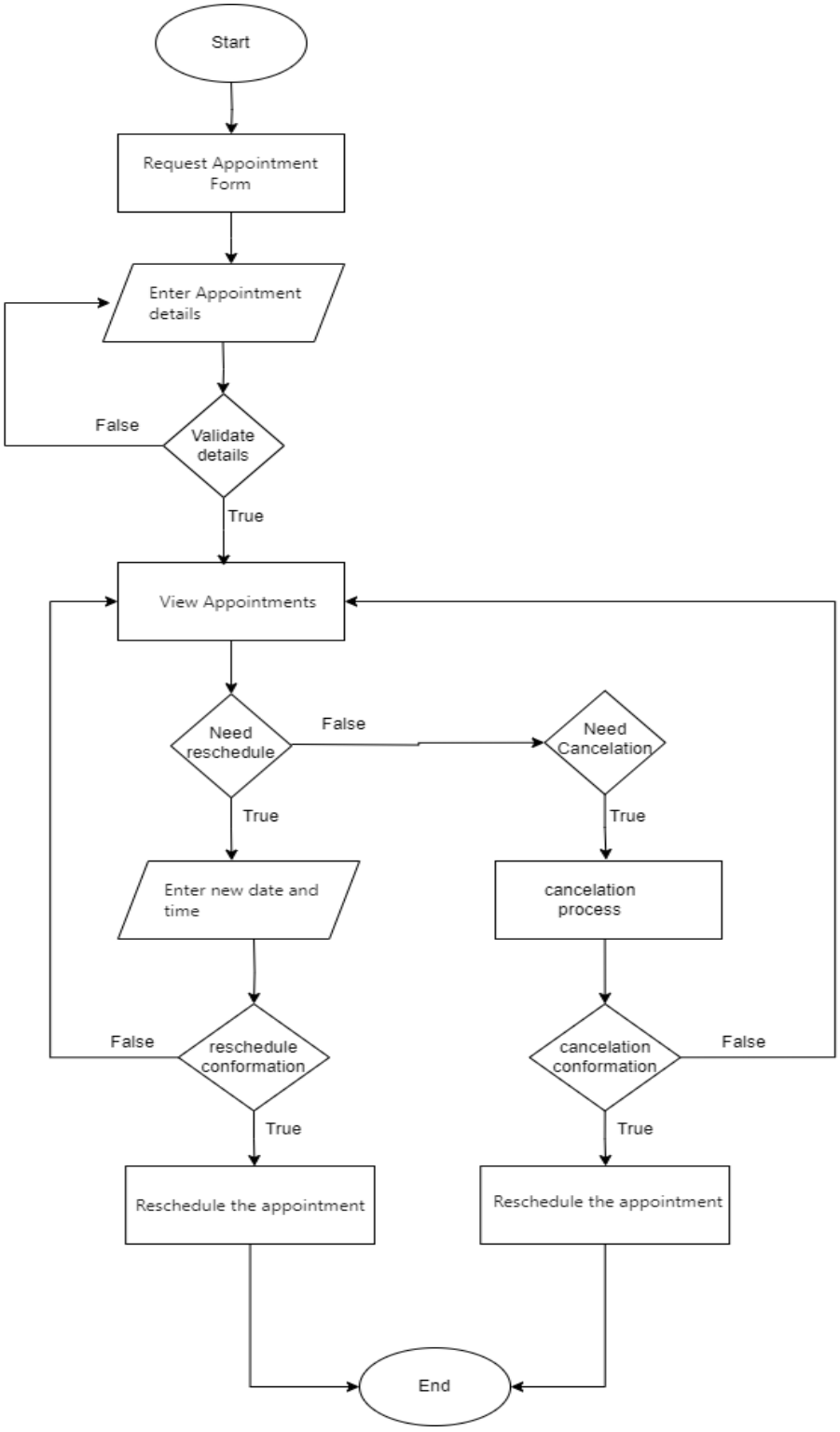
a. User selects the appointment to cancel.

b. Confirm the cancellation action.

5. c. If confirmed:
    - Change the appointment status to "canceled" in the database.
    - Provide acknowledgment of successful cancellation.
    - Remove the appointment from the user's scheduled appointments list.

End: The conclusion of the appointment management algorithm.

5.3.2 Flowcharts



5.3.3 Pseudocode

```
BEGIN
  Request Appointment:
    Prompt user to schedule appointment
    Validate input data

  IF Data is Valid THEN
    Save appointment
    Display acknowledgment
  ELSE
    Display error message
    Allow retry

  View Appointments:
    Retrieve appointments
    Display details

  Reschedule Appointment:
    Prompt user to select appointment
    Prompt for new date and time
    Validate new date and time

  IF New Date and Time are Valid THEN
    Update appointment
    Display acknowledgment
  ELSE
    Display error message
    Allow retry

  Cancel Appointment:
    Prompt user to select appointment
    Confirm cancellation

  IF Cancellation Confirmed THEN
    Cancel appointment
    Display acknowledgment
  END IF

END
```

6.Package Management System Development Progress

IT22578396 - Dayarathne R.D.T.N.

Outline

Newton Electrical's package management system attracts the customers for the company. Package managers can easily add packages with specific details like package name, description, ID, services, and prices ect... He can also update package information as needed, such as for seasonal changes. And the package manager can delete the package from the system. The system allows package manager to generate reports about packages, providing useful insights for decision-making. This simplifies the process of managing packages and ensures customers get the services they need efficiently.

6.1 Features' Completion Level:

Finished Features:

- Implemented the Admin side package page
- Implemented Package adding
- Implemented validation parts in admin page
- Implemented package report generation functionality
- Implemented CRUD operation

Features Not Yet Finished:

- Implement Customer side package page
- Implement the package prices and discount calculation

6.2 SQL Queries Used (With a example):

```
import mongoose from "mongoose"

const packageSchema = new mongoose.Schema({

  packageName
  type:String,
    required: true,
  },
  packageId:{
    type:String,
    required:true,
    unique: true,
  },
```

```
description: {
   type: String,
   required: true,

},
service1:{
   type: String,
   required:true
},
service2:{
   type: String,
   required:true
},
service3:{
   type: String
},
service4:{
   type: String
},
service1Price:{
   type:Number,
   required:true
},
service2Price:{
   type:Number,
   required:true
},
service3Price:{
   type:Number
},
service4Price:{
   type:Number
},
monthlyPrice:{
   type:Number,
   required:true,
},
annualPrice:{
   type:Number,
   required:true,

},
      monthlyDiscount:{
```

```
    type:Number,
    required:true,
  },
  annualDiscount:{
    type:Number,
    required:true,

  },

  discountMonthly:{
    type:Number,
    required:true,

  },
  discountAnnual:{
    type:Number,
    required:true,
},{timestamps: true})
const Package = mongoose.model('Package',packageSchema) //this becomes users in DB
export default Package
```

- CREATE

  INSERT INTO Package (packageName, packageId, description, service1, service2, service3, service4, service1Price, service2Price, service3Price, service4Price, monthlyPrice, annualPrice, Annualdiscount, Monthlydiscount, discountedMonthly, discountedAnnual)

  VALUES (‘Repairs’, ‘P005’, ‘In this package, we offer prompt and efficient repair services’, ‘malfunction repair’, ' breakdown repair ', ‘Equipment repair’, ‘Appliance Repair’, 100, 700, 200, 400, 8800, 26000, 5, 10, 600, 1800);

- UPDATE
  UPDATE Package
  SET packageName = 'Electronic repairs’', service2Price = 800
  WHERE packageId = 'P005’';
- DELETE
  DELETE FROM Package
  WHERE packageId = 'P005';

- READ
  - Read all packages
  SELECT * FROM Package;

  -Read a specific package by packageId
  SELECT * FROM Package WHERE packageId = 'P005';

6.3 Algorithms, Flowcharts, Pseudocode:

6.3.1 Algorithms

1.Start: The beginning of the Package management algorithm.

2.Request Package Action: Prompt the user to add a package.
3.Add a package :
a. User inputs package details including packageName, packageId, description, service1, service2, service3, service4, service1Price, service2Price, service3Price, service4Price, monthlyPrice, annualPrice, Annualdiscount, Monthlydiscount, discountedMonthly,discountedAnnual

b. Validate the input data for completeness and correctness.
c. If the data is valid:
- Insert the Package details into the database.
-Provide acknowledgment of successful package submission.
d. If the data is invalid:
- Display an error message indicating the required fields or format of data.
-Allow the package manager to retry adding a package.

4.View Package:
a. Package manager can access the package details.
b. Retrieve package details from the database.
c. Display package details to the package manager for review and management.

4.Update Package:
a. Package manager can update Package details if needed.
b.Prompt the user to Update package details.

c. Validate the updated details for completeness and correctness.

d. If the data is valid:

-Update the Package with new details into the database.

-Provide acknowledgment of successful package updation.

e. If the data is invalid:

- Display an error message indicating the required fields or format of data.

-Allow the package manager to retry updating package details.

4.Delete a package:

a. Package manager selects a package to delete.

b. Confirm the deletion action.

d. If confirmed,

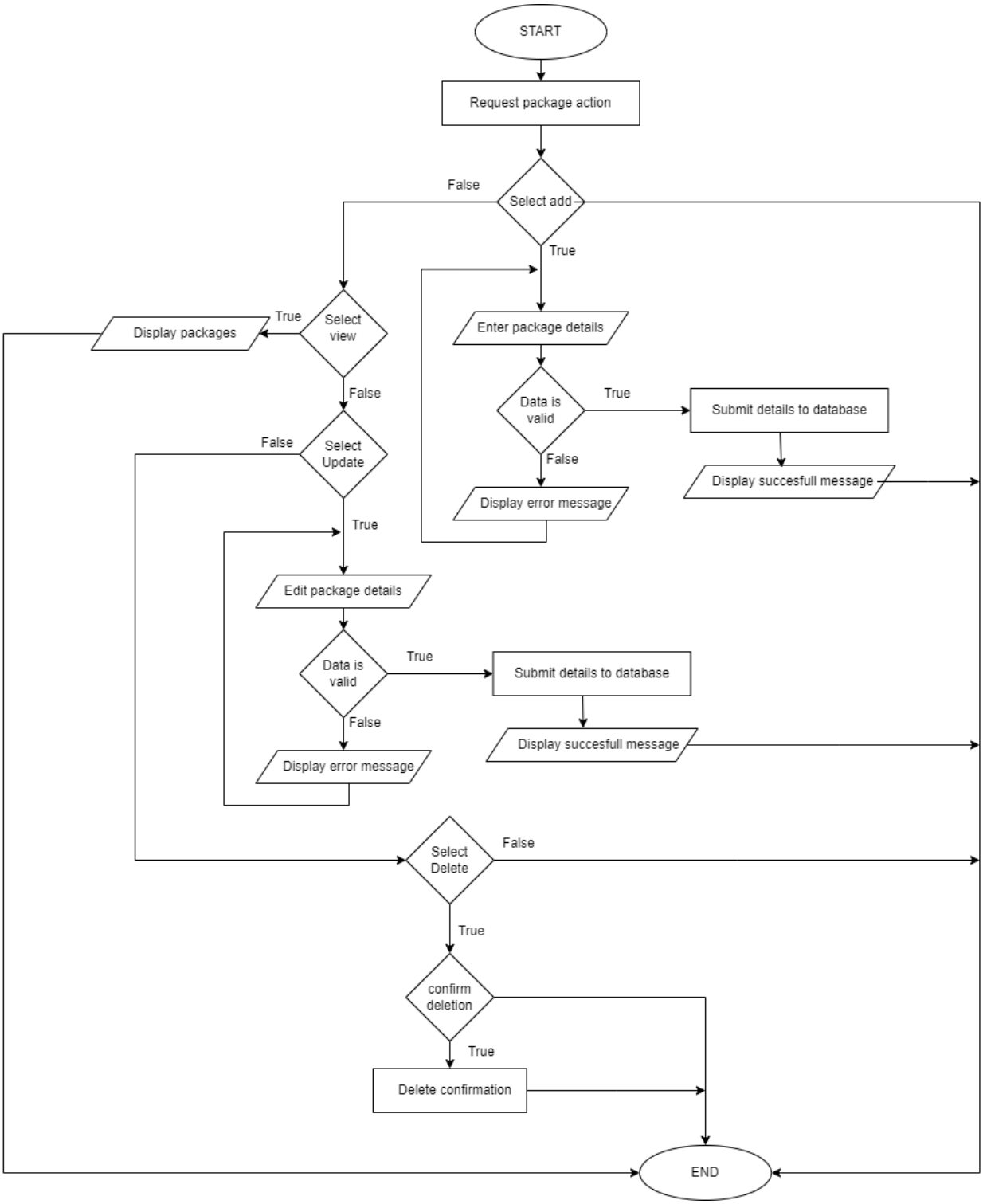
-delete the selected package from the database.

5.End: The conclusion of the package management algorithm.

6.3.2 Pseudocode

```
BEGIN

    REQUESTS package action (add, view, update, delete)

    IF package action is add THEN
        PROMPT user for add package form ( packageName, packageID. price, ect)
        VALDATE input data ( fill the all required fields)
        IF package details valid THEN
            INSERT details into database
            DISPLAY " Successfully added"
        ELSE
            DISPLAY "Fill the all required fields" ALLOW user to retry adding
            package
        ENDIF

    ELSE IF package action is view THEN
        SELECT * FROM package
    ELSE IF package action update THEN
        PROMPT user to edit package details( packageName , PackageID, ect…)
```

```
        VALDATE input data ( fill the all required fields)
        IF package details valid THEN
            UPDATE  details into database
            DISPLAY " Successfully updated"
        ELSE
            DISPLAY "Fill the all required fields" ALLOW user to retry updating
            package
        ENDIF
    ELSE package action is delete THEN
        PROMPT for confirmation
        IF confirmed THEN
            DELETE package data from database
            DISPLAY "confirmation message"
            NAVIGATE to package page
        END IF

    END ID

END
```


START
Request package action
Select add
False
True
Select view
True
Display packages
False
Select Update
False
True
Enter package details
Data is valid
True
Submit details to database
Display succesfull message
False
Display error message
Edit package details
Data is valid
True
Submit details to database
Display succesfull message
False
Display error message
Select Delete
False
True
confirm deletion
True
Delete confirmation
END

7. Project Management System Development Progress -

IT22581716 - Perera M.M.D

Outline

The Project Management function within Newton Electricals' System serves as a pivotal component, seamlessly merging past and ongoing projects. Primarily overseen by a designated project manager, this module encompasses a spectrum of tasks aimed at enhancing project visibility and efficiency. The project manager's role encompasses adding new projects, categorizing them into distinct phases such as completed, ongoing, and future endeavors, and updating their statuses accordingly. This systematic approach fosters a structured representation of project history, encapsulating detailed insights into outcomes, encountered challenges, and key performance metrics, thus facilitating informed decision-making processes.

One of the salient features of the Project Management function is its provision of real-time updates on ongoing projects. By delivering transparent progress reports, the system empowers stakeholders with the requisite information to make informed decisions. This transparency not only fosters accountability but also streamlines communication channels, thereby optimizing resource allocation and timelines. Moreover, the integration of past project details with ongoing initiatives ensures that historical data serves as a valuable asset, guiding current endeavors towards successful outcomes.

The holistic integration of past and ongoing project details significantly augments overall operational efficiency. By leveraging historical insights, the system can adaptively allocate resources, optimize timelines, and enhance client communication protocols. This approach underscores the system's commitment to delivering high-quality electrical services in a dynamic and informed manner, thereby bolstering stakeholder satisfaction and organizational reputation.

Furthermore, the project manager assumes a multifaceted role within the system, extending beyond project oversight to encompass user management functionalities. In addition to overseeing project lifecycles, the project manager has the authority to create, edit, and manage projects within the system's framework. Notably, they possess the prerogative to remove projects, thereby ensuring that outdated or irrelevant information is promptly expunged from the system.

Moreover, the project manager serves as the custodian of project-related data, tasked with generating comprehensive project reports. These reports offer stakeholders a bird's eye view of the company's project landscape, detailing metrics such as the number of completed projects, ongoing initiatives, and comprehensive breakdowns of project activities. By providing

stakeholders with granular insights into project performance, these reports facilitate strategic decision-making and enable continuous improvement across organizational processes.

In summary, the Project Management function within the electrical service management system embodies a robust framework designed to streamline project workflows, enhance transparency, and drive organizational efficiency. By leveraging innovative features and functionalities, the system empowers stakeholders to navigate complex project landscapes with confidence, thereby ensuring the seamless delivery of high-quality electrical services in a dynamic and ever-evolving industry landscape. With an emphasis on finished features, ongoing projects, SQL queries utilized, and algorithms implemented, this paper describes the development of the Project Management System.

7.1 Features' Completion Level:

Finished Features:

- Completed admin side page which can only be viewed by the Project Management.
- Implemented a validation parts in admin page form.
- Additionally, implemented project count, pie chart, search functionality, and report generation in the admin dashboard.
- Almost completed customer side page viewable by admin as well as all users.

Features Not Yet Finished:

- Upgrading evaluating bar to more responsive and attractive way in customer side completed UI’s.

7.2 SQL Queries Used (With a example):

```
// models/ProjectModel.js

import mongoose from "mongoose"
const projectSchema = new mongoose.Schema({
   title:{
      type:String,
      required:true,
   },
```

```
    status:{
        type:String,
        default:"Ongoing",
    },

    description:{
        type:String,
        required:true,
    },

    duration:{
        type:String,
        required:true,
    },

    cost:{
        type:String,
    },
    image:{
        type:String,
    },

},{timestamps: true})
const Project = mongoose.model('Project',projectSchema)
export default Project
```

- Inserting a Project:

```
INSERT INTO Project (Title, Status, Description, Duration, Cost, Image)
VALUES ('Coffee Lab - Hatton', 'Previous', '10 Million worth rewiring of the Main Building of Coffee Lab.  Panel boards newly installed.', '6 Months', '50000000', '1713315953037maxresdefault.jpg');
```

- Finding Project:

```
SELECT * FROM Project WHERE Status = 'Previous';
```

- Updating a Project(e.g., changing cost):

```
UPDATE Project SET Cost = '80000000' WHERE Title = 'Coffee Lab - Hatton';
```

- Deleting a Project:

```
DELETE FROM Project WHERE Title = 'Coffee Lab - Hatton';
```

7.3 Algorithms, Flowcharts, Pseudocode:

7.3.1 Algorithms

1. Start - The initialization of the Project Management function within Newton Electricals' System.
2. Login to the system.
   a. Input login credentials (Email and Password).
   b. Validate credentials.
   c. If credentials are valid - Navigate to Home page.
   d. If credentials are invalid - Display "Invalid Username and Password".
   - Navigate to Login page.
4. Click Admin Dashboard button.
   a. Navigate to Admin Dashboard.
5. Click Projects button.
   a. Display Project Manager UI.
   b. On this page, the project census pie chart on Previous and Ongoing will appear first.
6. Click Add Project button.
   a. Display Add Project form.
   b. Input project details including Title, Status, Description, Duration, Cost, and Image.
   c. Validate the input data for completeness and accuracy.
   d. If the data is valid - Insert the project data into the system's database.
   - Provide confirmation of successful project addition.
   e. If the data is invalid - Display an error message indicating the missing or incorrect fields.
   - Allow the Project Manager to retry providing project details.
   f. Submit form.
6. Display the table with Updated Project Details.
   a. The Project Manager updates the status of ongoing projects.
   b. Search the Title and select the new Status (Completed or Ongoing).
   c. Validate the input data for correctness.
   d. If the data is valid - Update the project Status in the database.
   - Provide confirmation of successful Status update.
   e. If the data is invalid - Display an Error Message indicating the Title or Status.
   - Allow the Project Manager to retry updating the project Status.

7. Display the Delete button.
   a. A notification will appear to confirm whether or not to delete these details and cancel.
   b. If the click Cancel - The details in the table are not deleted.
   c. If the click Delete - The Project Manager can delete irrelevant details as needed and the details in the table are deleted.
8. Display Generate Project Reports button.
   a. The Project Manager generates comprehensive project reports.
   b. Access project data from the database.
   c. Compile metrics such as Completed Projects, Ongoing initiatives, and project activities breakdown.
   d. Format the data into a structured report format.
   e. Provide stakeholders access to view and analyze project reports for informed decision-making.
9. View Project Details.
   a. Customers and authorized employees can view project details by access their dedicated UI.
10. End - The conclusion of the Project Management function algorithms within Newton Electricals' System.

7.3.2 Flowcharts

7.3.3 Pseudocode

```
BEGIN
  Project Manager Login to the system
INPUT
  Login Credentials (Email and Password)
VALIDATE
  Credentials against stored data
IF
 Credentials are valid
THEN CHECK
 User role
IF
 User role is administrator
THEN GRANT
 Access to Admin Dashboard
CLICK
  Project button
DISPLAY
  Project Manager UI
DISPLAY
  The project census pie chart on Previous and Ongoing
DISPLAY
  Add Project button
CLICK
  Add Project button
PROMPT
  For required project details (title, status, description, duration, etc.)
VALIDATE
  Input data (the description should be less than five hundred words, cost column should contain
  only numbers)
IF
  Project detail is valid
SUBMIT
  Form

THEN INSERT
   Data into the database
END IF
THEN DISPLAY
```

Project details table
IF CHOOSES
To update details
THEN CLICK
Update button
PROMPT
For Project Manager to enter new information (e.g., cost, duration, etc.)
VALIDATE
Updated user information
UPDATE
Data in the database
END IF
THEN DISPLAY
Delete button
CLICK
Delete button
DISPLAY
Display the notification to confirm and cancel the deletion
IF CHOOSE
To delete
CLICK
Delete
THEN
The details in the table are not deleted
ELSE CHOOSE
To cancel
CLICK
Cancel
THEN
The details in the table are deleted
END IF
IF CHOOSE
To view details

NAVIGATE
Customers and authorized employees can view project details by access their dedicated UI.

END IF
ELSE
DISPLAY

Invalid Username and Password
NAVIGATE
To Login page
END IF
END IF
END

7.4 Final Completion To-Do List:

- Upgrading cards to more responsive and attractive way in customer side completed UI's.

8. Order Management System Development Progress

IT22322876 – Madhubhashana M D H P

Outline

Newton Electrical's order management system is designed in a way which greatly enhances the customer's online shopping experience. All key functionalities of the OMS are only available to authenticated users. Customers can add their desired products to the wishlist, add products to the cart and checkout, place orders directly, make payments, and access after-sales services. Order managers do the review, report generation, customer activity monitoring, and solving of unresolved disputes / complaints. With the ability to handle large volumes of orders, proper integration with delivery tracking services and payment processors, and real time notifications system, OMS enhances the overall operational effectiveness of the whole system.

8.1 Features' Completion Level:

Finished Features:

- Implemented OMS features of store and product description pages.
- Implemented wishlist and shopping cart CRUD functionality.
- Implemented place order and order summary pages.

Features Not Yet Finished:

- Implement Customer side my orders, order details, delivery tracking and after sales dispute raising/complaint pages.
- Implement Admin side (order manager) of the OMS.
- Integrate with delivery tracking services and payment processors such as PayPal and Stripe.
- Integrate with real-time on-site notification system.
- Implement alerts functionalities for sending mobile and email notifications.

8.2.1 Database Schemas

Order Schema

```
import mongoose from "mongoose"

const orderSchema = new mongoose.Schema({

        orderType: {
        type: String,
    enum: [‘PRODUCT’, ‘SERVICE’],
    default: ‘PRODUCT’
        required: true,
  },

  placedBy: {
    type: Schema.Types.ObjectId,
    ref: ‘User’
  }, // represents the customer

  fulfilledBy: {
    type: Schema.Types.ObjectId,
    ref: ‘User’
  }, // represents the supplier

  total: {
    type: Number,
    required: true
  },

  discountedTotal: {
    type: Number,
  },

  products: [{
    type: Schema.Types.ObjectId,
    ref: ‘ProductOptionItem’
  }],
```

```
    services : [{
        type: Schema.Types.ObjectId,
        ref: ‘Service’
    }],

    status: {
        type: String,
        enum: [‘PROCESSING’, ‘AWAITING_PAYMENT’, ‘AWAITING_SHIPMENT’,
‘AWAITING_COLLECTION’, ‘AWAITING_FULFILLMENT’, ‘SHIPPED’, ‘COMPLETED’,
‘CANCELED’],
        required: true
    },

    payments: [{
        type: Schema.Types.ObjectId,
        ref: ‘Payment’
    }],

    delivery: {
        type: Schema.Types.ObjectId,
        ref: ‘Delivery’
    },

    disputes: [{
        type: Schema.types.ObjectId,
        ref: ‘Dispute’
    }],

    remarks: {
        type: String
    },

}, {timestamps: true})

const Order = mongoose.model('Order', orderSchema)
export default Order
```

productOptionItemSchema

```
import mongoose from "mongoose"

const productOptionItemSchema = new mongoose.Schema({


        product: {

                type: Schema.Types.ObjectId,

                ref: ‘Product’

        }, // populate selected product from Product model to get up-to-date information about the product


        quantity: {

                type: Number,

                required: true

        }, // selected product quantity


        options: {

                type: String[],

                required: true

        }, // selected options such as type, color, power etc.


        itemTotal: {

                type: Number,
```

```
            required: true
        }, // total price based on quantity and options of one selected product

    discountedItemTotal: {
        type: Number
        }, // discounted total price per product
})

const ProductOptionItem = mongoose.model('Delivery', productOptionItemSchema),
export default ProductOptionItem
// This represents a product added to the cart or ordered.
// Helps to store quantities or options of a product that has been added to the cart.
```

deliverySchema

```
import mongoose from "mongoose"
 const deliverySchema = new mongoose.Schema({

        order: {
            type: Schema.Types.ObjectId,
            ref: 'Order'
        },

        deliveryMode: {
```

```
            type: ScheString,
            enum: [‘HOME_DELIVERY’, ‘STORE_PICKUP’]
            default: ‘HOME_DELIVERY’,
            required: true
        },

        source: {
            type: String,
            required: true
        },

        destination: {
            type: String,
            required: true
        },

   shippingService: {
            type: String,
            enum: [‘FREE_SHIPPING’, ‘STANDARD_SHIPPING’,
‘PRIORITY_SHIPPING’]
        },

        deliveryUpdates: [{
            type: Schema.Types.ObjectId,
```

```
        ref: ‘DeliveryUpdate’
    }],

}, {timestamps: true})

const Delivery = mongoose.model(‘Delivery’, deliverySchema),
export default Delivery
```

deliveryUpdateSchema

```
import mongoose from "mongoose"
const deliveryUpdateSchema = new mongoose.Schema({

    delivery: {
        type: Schema.Types.ObjectId,
        ref: ‘Delivery’
    },

    updateId: {
        type: String,
        required: true
    }

    title: {
        type: String,
```

```
        required: true
    },

    description: {
        type: String
    },

    pkgLocation: {
        type: String,
        required: true
    },

    status: {
        type: String,
        enum: ['PKG_LEAVING_ORIGIN', 'PKG_LEFT_ORIGIN', 'PKG_HANDED_OVER', 'PKG_IN_TRANSIT', 'PKG_REACHING_DESTINATION', 'PKG_DELIVERED', 'PKG_LOST', 'PKG_RETURN'],
        required: true
    },

    career: {
        type: String,
        required: true
```

```
    },

}, {timestamps: true})

const DeliveryUpdate = mongoose.model('DeliveryUpdate', deliveryUpdateSchema),

export default DeliveryUpdate
```

paymentSchema

```
import mongoose from "mongoose"

const paymentSchema = new mongoose.Schema({

    order: {
        type: Schema.Types.ObjectId,
        ref: 'Order'
    },

    paymentId: {
        type: String,
        required: true
    },

    payMode: {
```

```
        type: String,
        enum: [‘PAY_FIRST’, ‘CASH_ON_DELIVERY’],
        default: ‘PAY_FIRST’,
        required: true
    },

    type: {
        type: String,
        enum: [‘ONE_TIME’, ‘INSTALLATION’],
        default: ‘ONE_TIME’,
        required: true
    },

    method: {
        ype: String,
        enum: [‘CARD’, ‘CHEQUE’, ‘EFT’, ‘VIRTUAL_WALLET’,
‘WIRE_TRANSFER’],
        required: true
    },

    amount: {
        type: Number,
        required: true
    },
```

```
        status: {
                type: String,
                enum: [‘PROCESSING’, ‘COMPLETED’, ‘CANCELED’, ‘FAILED’, ‘REJECTED’],
        }

}, {timestamps: true})

const Payment = mongoose.model(‘Payment’, paymentSchema),
export default Payment
```

disputeSchema

```
import mongoose from "mongoose"
const disputeSchema = new mongoose.Schema({

        order: {
                type: Schema.Types.ObjectId,
                ref: ‘Order’
        },

        disputeId: {
```

```
        type: String,
        required: true
    },

    category: {
        type: String,
        enum: ['NOT_AS_EXPECTED', 'WRONG_SPECIFICS', 'ITEM_DAMAGED',
'NOT_RECEIVED', 'SERVICE_INCOMPLETE', 'SERVICE_ISSUE',
'TECHNICIAN_ISSUE', 'SUPPLIER_ISSUE', 'CARRIER_ISSUE'],
        required: true
    },

    title: {
        type: String,
        required: true
    },

    description: {
        type: String,
        required: true
    },

    evidence: {
        type: String[],
```

```
        required: true
    },

    status: {
        type: String,
        enum: ['AWAITING_SUPPLIER_RESPONSE', 'SUPPLIER_RESPONDED', 'EVIDENCE_UPDATED', 'UNSOLVED', 'SOLVED'],
    }

    solution: {
        type: String
    }

}, {timestamps: true})

const Dispute = mongoose.model('Dispute', disputeSchema),
export default Dispute
```

Relationships with userSchema

```
import mongoose from "mongoose"

const userSchema = new mongoose.Schema({

    …
```

```
shoppingCartProd: [{
        type: Schema.Types.ObjectId,
        ref: 'ProductOptionItem'
}], // populate products added to cart from ProductOptionItem model

wishlistProd: [{
        type: Schema.Types.ObjectId,
        ref: 'Product'
}],
// populate products added to wishlist from Product model
// quantity and product options doesn't matter in the wishlist

shoppingCartServ: [{
        type: Schema.Types.ObjectId,
        ref: 'Service'
}], // populate services added to cart from Service model

wishlistServ: [{
        type: Schema.Types.ObjectId,
        ref: 'Service'
}], // populate services added to wishlist from Service model
```

```
    orders: [{
        type: Schema.Types.ObjectId,
        ref: 'Order'
    }], // populate my orders from Order model
    …
})

const User = mongoose.model('User', userSchema),
export default User
```

8.2.2 SQL Queries Used (With a example):

- Customer placed a new order for a product and saved to pay later

```
INSERT INTO Order (orderId, orderType, total, discountedTotal, status, placedBy)
VALUES ('OP_615', 'PRODUCT', 16500, 14356, 'AWAITING_PAYMENT', 'U_1453');
```

- Customer placed a new order for a service and paid

```
INSERT INTO Order (orderId, orderType, total, discountedTotal, status, placedBy)
VALUES ('OS_735', 'SERVICE', 55000, 49500, 'AWAITING_FULFILLMENT', 'U_1213');
```

- Customer paid a saved order (for a product), update status

UPDATE Order
SET status = ‘AWAITING_SHIPMENT’
WHERE orderId = ‘OP_536’;

- Order manager get notified about a fraud, cancels the order

  UPDATE Order
  SET status = ‘CANCELED’
  WHERE orderId = ‘OP_539’;

- Order manager retrieves a list of all orders for a time range

  SELECT * FROM Order;

  WHERE placedAt BETWEEN '2024-03-20' AND '2024-04-20';

- Customer retrieves a list of all orders aka ‘My Orders’

  SELECT * FROM Order;

  WHERE placedBy = 'U_1615';

- Customer retrieves a list of added product details from shopping cart

  SELECT * FROM ShoppingCartItems;

  WHERE addedBy = 'U_1615'

- Customer retrieves a list of added product titles from wishlist

  SELECT productTitle FROM WishlistItems;

  WHERE addedBy = 'U_1513'

- Customer updates details of an added product in shopping cart

  UPDATE ShoppingCartItems

SET quantity = 5, options = ‘color: platinum, type: with-neon’
WHERE itemId = ‘SCI_453’;

8.3.1 Algorithms

1. Start

2. Logged-in user enter product quantity

3. Select product options (color, type etc.)

4. If user chooses add to cart:

   a. Insert data into the database

5. If user chooses add to wishlist:

   a. Insert data into the database

6. If user chooses buy now:

   a. Proceed to order confirmation page.

   b. Prompt for required information (shipping address, delivery mode, order notes etc.)

   c. Validate input data, calculate OMS related discounts and proceed to order summary page.

   d. Prompt to continue to payment or pay later.

   e. If user chooses pay later, insert order data to database and redirect to my orders page.

   f. If user chooses to continue to payment, proceed to payment page.

   g. Prompt for required information (payment method, type, mode and details etc.)

   h. Validate input data against rules of the chosen payment method.

   i. If input data is valid:

      - Select a suitable payment processor.

- Process the payment.

- Notify user about the status of the payment.

j. If input data is invalid:

- Display an informative error message.

- Limit re-entry to 5 times.

- If attempts exceed the limit, prompt the user to verify captcha.

- If the user couldn’t solve captcha for 20 times, automatically limit the the ability to make payments for 1 day.

7. If a logged-out user or an unregistered user try to add a product to cart, add to wishlist or place an order:

a. Display ‘Login required’ and navigate to login page.

8. If a logged-in user chooses ‘all orders’ from the filter option in my orders page.

a. Fetch all orders and display them.

9. If a logged-in user chooses to track his order:

a. Connect to order tracking API.

b. Fetch latest details regarding the package and display them.

10. If a logged-in user chooses to cancel a paid order:

a. Check the real-time order status.

b. If the order is already shipped, notify the user and navigate to tracking page of the particular order.

c. If the order is not yet shipped:

- Notify the supplier regarding the cancellation.

- Wait 3 days for the supplier to respond.

- If supplier approves the request, notify the user and process a refund.

- If supplier rejects the request, wait 3 more days for the order manager ro respond.

- If supplier doesn’t respond for 3 days, automatically process a refund.

11. End.

8.3.2 Pseudocode

```
BEGIN Order Management

    FUNCTION ProcessOrderAction(action, input)
        IF  action is ‘ADD_TO_CART’ THEN
            WHILE i < input.length
                INSERT INTO ShoppingCartItems(itemData, addedBy)
                VALUES(input[i], activeUser)
        ELSE IF action is ‘ADD_TO_WISHLIST’ THEN
        WHILE i < input.length
                INSERT INTO WishlistItems(itemData, addedBy)
                VALUES(input[i], activeUser)
        ELSE IF action is ‘BUY NOW’ THEN
            PROCEED to order confirmation page
            PROMPT for required information (shipping address, delivery mode,
notes etc.)
            VALIDATE input data
            CALCULATE discounted price for the order
            NAVIGATE to order summary page
```

```
            PROMPT to pay now or pay later
            IF user action is pay now THEN
                handlePayment()
            ELSE IF user action is pay later THEN
                INSERT INTO Orders(orderData, status, placedBy)
                VALUES(inputData, 'AWAITING_PAYMENT', activeUser)
            END IF
        END IF
END FUNCTION

FUNCTION handlePayment(input)
        IF input data is not null THEN
            VALIDATE input data against payment method rules
            IF input data is valid THEN
                SELECT a payment processor
                PROCESS the payment
                NOTIFY user about the status
            ELSE
                DISPLAY 'Enter valid payment information'
                LIMIT retry to 5 times
                IF attemps exceed the limit THEN
                    WARN the user
                    REQUEST to solve captcha
```

```
                    IF user fails to solve 20 times THEN
                        LIMIT user from making payments for 1 day
                    END IF
                END IF
            END IF
        END FUNCTION

        IF user is not logged in or unregistered THEN
            DISPLAY ‘Login required’
            NAVIGATE to login page
        END IF

        IF user selects ‘All orders’ from my orders page THEN
            FETCH all orders
            DISPLAY order data
        END IF

        IF user action is track order THEN
            CONNECT to order tracking API
            FETCH latest information regarding the package
            DISPLAY delivery data
        END IF
```

```
IF user chooses to cancel a paid order THEN
        IF order status is ‘SHIPPED’ THEN
                NOTIFY the user
        NAVIGATE to tracking page
        ELSE
                NOTIFY the supplier regarding the cancellation
                WAIT 3 days for the supplier to respond
                IF supplier approved THEN
                        CANCEL the order
                PROCESS a refund
                ELSE IF supplier rejected THEN
                        WAIT 3 days for the order manager to respond
                ELSE IF no response for 3 days THEN
                        CANCEL the order
                PROCESS a refund automatically
                END IF
        END IF
END IF



FUNCTION main(action, input)
        ProcessOrderAction(action, input)
END FUNCTION
```

main(action, input)

END Order Management