Sri Lanka Institute of Information Technology

Information Technology Project (IT2080)

ITP24J_B03_04

BidMaster
Auction System

Assignment 2

80% Progress Review

Submitted by:

	Name with Initials (Surname first)	Registration Number	Contact Phone Number	Email
1.	Hettiarachchi H.A.S.L	IT22246332	0759523693	it22246332@my.sliit.lk
2.	Athukorala H.H.B	IT22066770	0705790678	it22066770@my.sliit.lk
3.	Pihillegedara S.N.M	IT22238580	0714446216	it22238580@my.sliit.lk
4.	Viduranga S.P.S	IT22215192	0702340694	it22215192@my.sliit.lk
5.	Thalangama A.I	IT22235138	0702874476	it22235138@my.sliit.lk
6.	Sanjana K.G.T.S	IT22224170	0712568275	it22224170@my.sliit.lk
7.	Samaranayake W.M.R.L	IT22606174	0766695805	it22606174@my.sliit.lk
8.	Chandraguptha H.A.T	IT22091970	0701722473	it22091970@my.sliit.lk

Page of Contents

Project Title	BIDMASTER Auction System
Repository Link	https://github.com/sandunlak/BIDMASTER/tree/Hirun

1.User Stories

Function	**User story**
Delivery management	“As a seller I want to add sold items to the system, so that I can show my orders to bidders and shipping providers”
	“As a seller I want to add delivery person to the system, so that I can handle delivery with more secure”
	“As a shipping service provider, I want to update delivery states so that I can show delivery states to the sellers and bidders”
	“As a Shipping provider I want to provide shipping cost based on item details and destination So that Buyers can track the states of their orders.”
	“As a Shipping provider I want to generate shipping labels for items being shipped to Sellers can print and attach them to packages for delivery”

Function	**User Story**
Bidder	“As a bidder, I want to view the current highest bid on an item, so that I can decide how much to bid.”
	“As a bidder, I want to place a new bid on an item, so that I can participate in the auction”
	“As a bidder, I want to receive a confirmation after successfully placing a bid, so that I know my bid was accepted”
	“As a bidder, I want not to be able to bid, If I’m going to outbid so that I can decide whether to place a higher bid”
	“As a bidder, I want to see a countdown timer for the auction so I know when it will end.”
	“As a bidder, I want to be notified if I win an auction, so that I can proceed with the payment process.”

Function	**User Story**
Seller	As a seller, I want to be able to register an account on the platform, providing necessary details such as my name, contact information, and preferred payment methods, so that I can start selling items
	As a seller, I want to create new auction listings for items I wish to sell, specifying details such as item description, starting price, auction duration, and any special terms or conditions, to attract potential buyers
	As a seller, I want to manage my active auction listings, including editing item details, updating auction end times, and closing auctions early if necessary, to maintain control over my sales
	As a Seller, I want to Register to upcoming auctions. So that I can enter item listings
	As a seller, I want to manage my Account, so that I can update my profile and details.
	As a seller, I want analytics about the auctions and items, so that I can use my strategies.
Auction handling Manager	As an auction registration manager, I want to monitor auction listings to ensure they comply with platform policies and guidelines, intervening if necessary to resolve disputes or remove inappropriate listings
	As an auction registration manager, I want to assist sellers and buyers with any issues they encounter during the auction process, providing support through clear communication and timely resolution of disputes
	As an auction registration manager, I want to oversee and approve new seller registrations, ensuring they provide accurate and valid information to maintain the integrity of the platform
	As an Auction handling manager, I want to handle auction items.
	As an auction handling manager, I want to add new auction, update auctions and delete auctions so that users can register to auctions.

Function	**User Story**
Advertising Handler	"As an Advertising Handler, I want to record advertisements accurately, so that all critical information is captured for each ad."
	"As an Advertising Handler, I want to update advertisement details, so that any necessary changes are reflected, and the ad remains current."
	"As an Advertising Handler, I want to delete incorrect or obsolete advertisements, so that only relevant and valid ads are displayed."
	"As an Advertising Handler, I want to maintain a central database for advertiser profiles, so that advertiser management is streamlined."
	"As an Advertising Handler, I want the option to remove advertiser details if they decide to discontinue their services, so that the database remains relevant and compliant."
	"As an Advertising Handler, I want the option to update advertiser details according to their requests, so that their information is kept accurate and up to date."

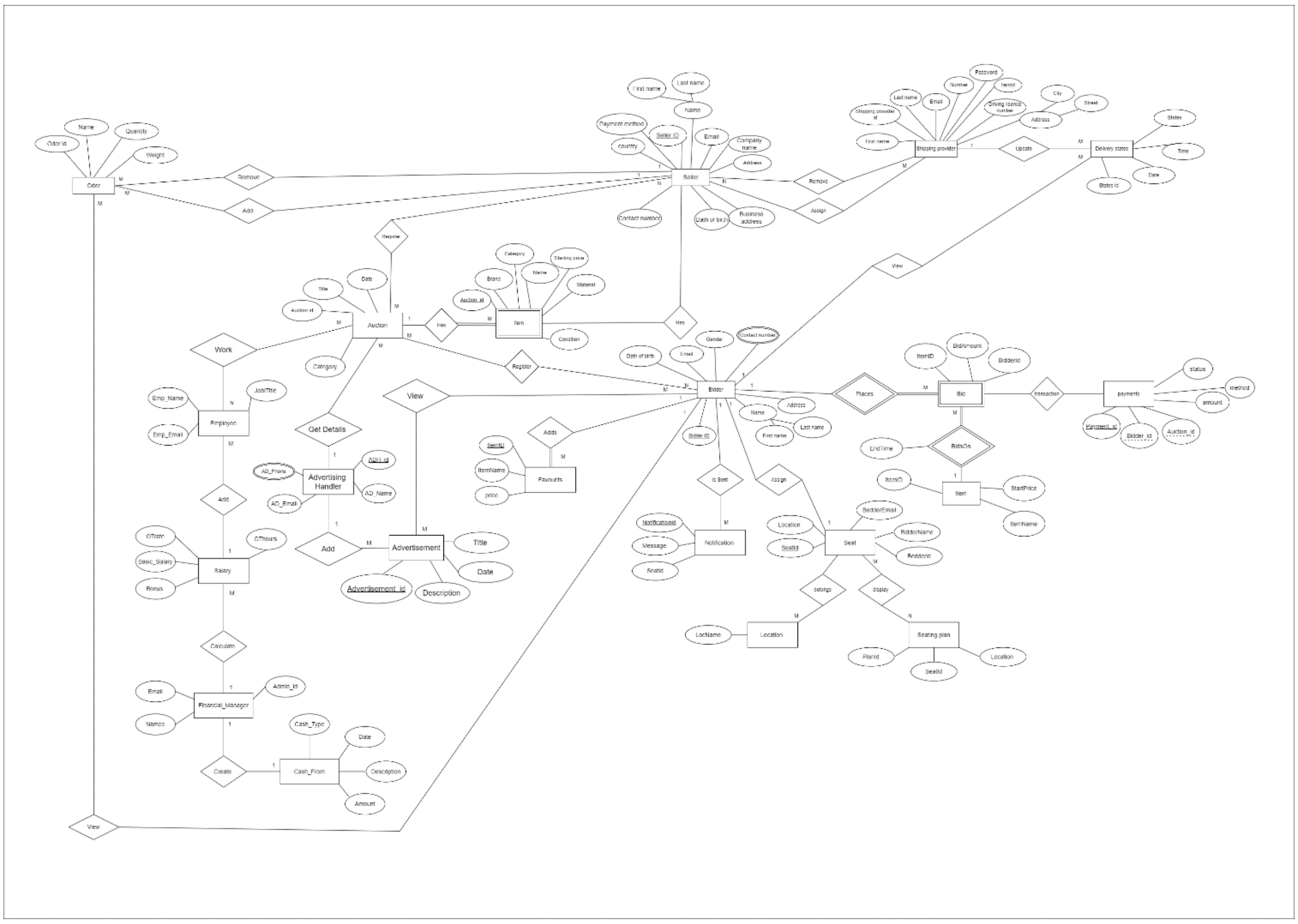
Function	**User Story**
User Registration management	“As a user, I want to sign up by providing my details, so that I can create an account on the auction system.”
	“As a registered bidder, I want to view available auction items, so that I can register and participate in auctions.”
	“As a registered seller, I want to create and manage auction listings, so that I can sell items on the platform.”
	“As a bidder, I want to view and update my profile details, so that I can keep my information up to date.”
	“As a bidder, I want to view the auctions I’ve registered for, so that I can track my participation.”

Function	**User story**
	As a financial manager, I want to be able to calculate salaries based on job roles so that I can ensure employees are paid accurately.
	As a financial manager, I want to add and store basic salary information for each employee so that I can calculate their total compensation.
	As a financial manager, I want to input OT (overtime) hours and OT rates ,bonuses, EPF&ETF deductions for each employee correctly
	As a financial manager, I want to be able to update and delete salary details, such as basic salary, OT hours, OT rates, and bonus, so that I can ensure all compensation information is current
	As a financial manager, I want to create a petty cash form to track small expenses and ensure accurate accounting of petty cash usage.
	As a financial manager, I want to generate and manage a balance sheet that summarizes all income and expenses so that I can monitor the financial health of the auction system.
	As a financial manager, I want to display all income and expenses in a table on the website so that stakeholders can easily view the financial data.

Function	**User Story**
Payment Gateway Integration	"As a seller, I want to process payments through the payment gateway for shipping services, so that I can ensure secure payment to the shipping providers."
	"As a buyer, I want to pay for items and shipping fees through the payment gateway, so that I can track the payment status securely."
	"As a payment gateway service provider, I want to confirm payment for items and shipping services, so that I can release the funds to sellers and shipping providers."
	"As a payment gateway provider, I want to track the status of payments and delivery milestones, so that I can ensure timely processing of payments once delivery is confirmed."
	"As a shipping provider, I want to receive payments automatically upon delivery confirmation, so that I don't have to manually follow up on payments."

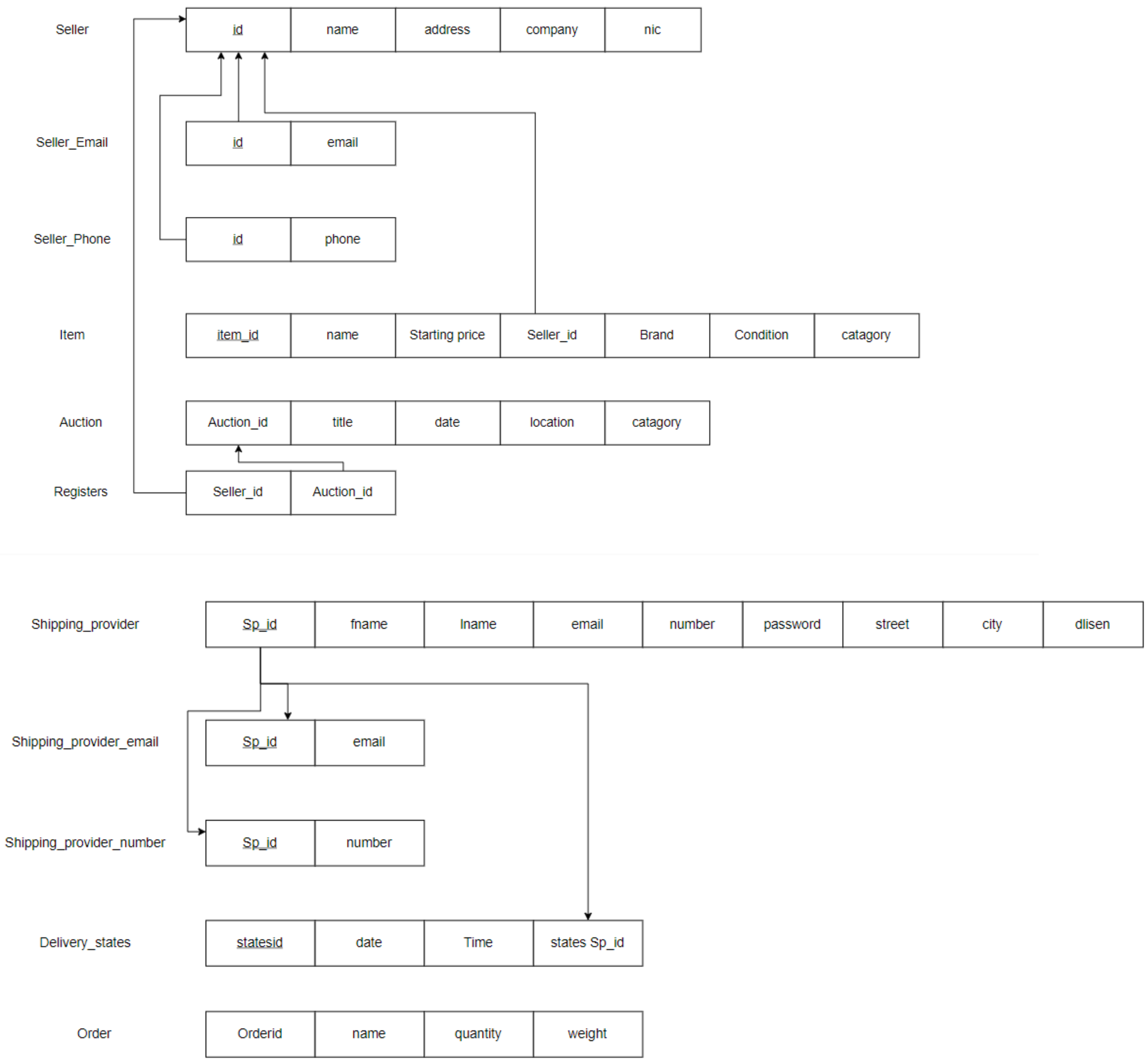
Auction event handling	User story
	“As a seating manager, I want the system to assign seats to registered bidders.”
	“As a seating manager, I want the system to generate a visual seating plan.”
	“As a seating manager, I want the system to send automated emails to bidders with their seat numbers.”
	“As a seating manager, I want to make adjustments to the seating plan.”

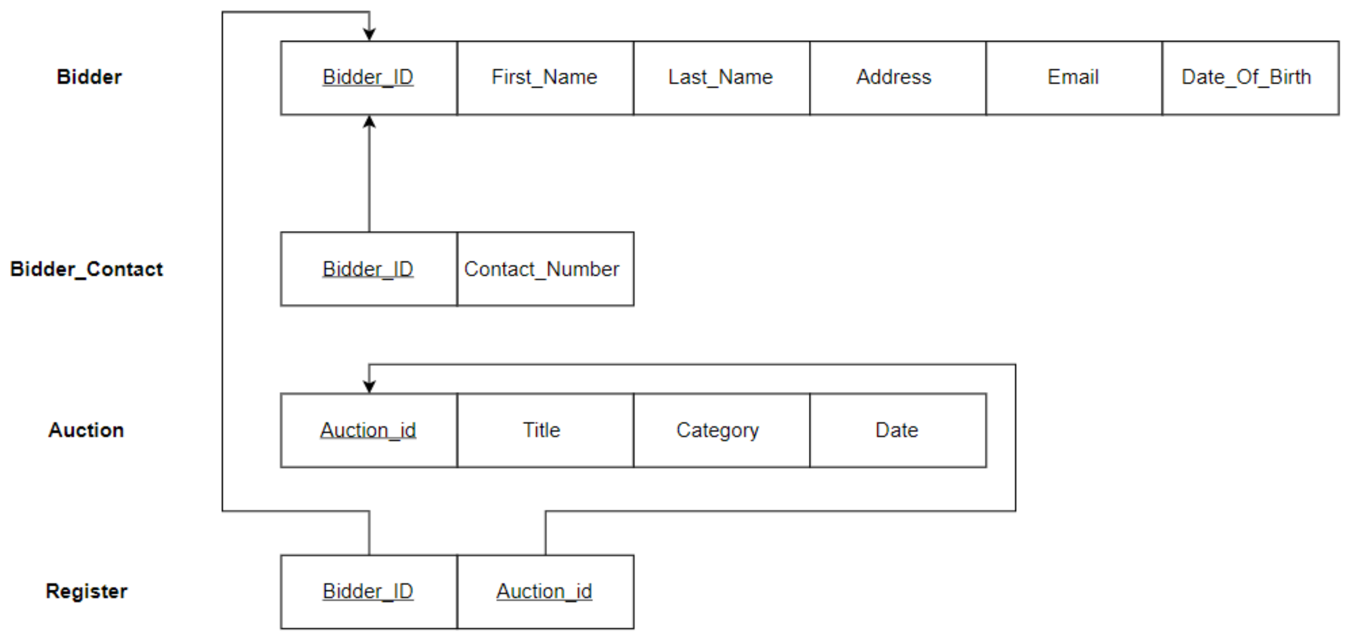
ER Diagram

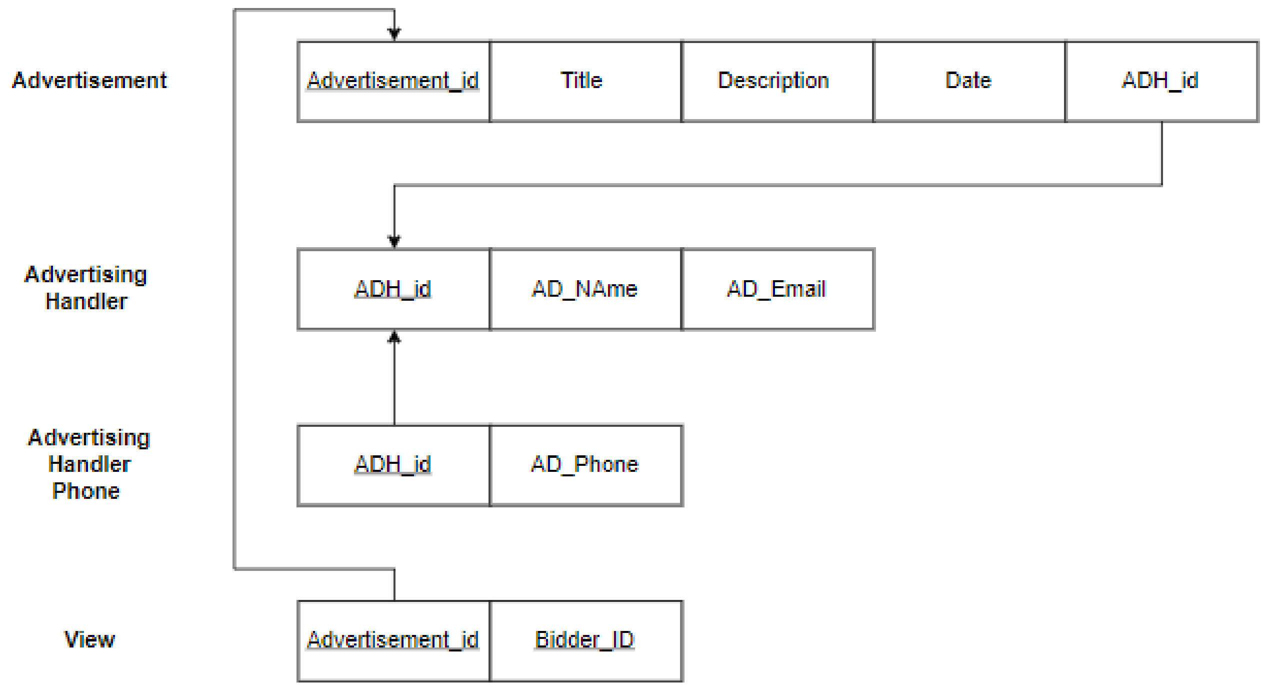


Oder
Oder id
Name
Quantity
Weight
Remove
Add
View
Seller
First name
Last name
Name
Payment method
Seller ID
Email
Company name
country
Address
Contact number
Dath of birth
Business address
Remove
Assign
Shipping provider
Shipping provider id
First name
Last name
Email
Number
Password
Itemid
Driving lisence number
Address
City
Street
Update
Delivery states
States
Time
Date
States id
Register
Auction
Auction id
Title
Date
Category
Has
Item
Auction_id
Brand
Category
Name
Starting price
Material
Condition
Has
Register
View
Bidder
Dath of birth
Email
Gender
Contact number
Address
Name
First name
Last name
Bidder ID
Places
Bid
ItemID
BidAmount
BidderId
transaction
payments
status
method
amount
Payment_id
Bidder_id
Auction_id
BidsOn
EndTime
Item
ItemID
StartPrice
ItemName
Adds
Favourits
ItemID
ItemName
price
Is Sent
Notification
NotificationId
Message
SeatId
Assign
Seat
Location
SeatId
BedderEmail
BidderName
Bedderid
belongs
display
Location
LocName
Seating plan
PlanId
SeatId
Location
Work
Employee
Emp_Name
Emp_Email
JobTitle
Add
Salary
OTrate
Basic_Salary
Bonus
OThours
Calculate
Financial_Manager
Email
Namee
Admin_Id
Create
Cash_From
Cash_Type
Date
Description
Amount
Get Details
Advertising Handler
AD_Phone
ADH_id
AD_Name
AD_Email
Add
Advertisement
Title
Date
Advertisement_id
Description
View

Normalized schema




Bidder
Bidder_ID
First_Name
Last_Name
Address
Email
Date_Of_Birth
Bidder_Contact
Bidder_ID
Contact_Number
Auction
Auction_id
Title
Category
Date
Register
Bidder_ID
Auction_id


Advertisement
Advertisement_id
Title
Description
Date
ADH_id
Advertising Handler
ADH_id
AD_NAme
AD_Email
Advertising Handler Phone
ADH_id
AD_Phone
View
Advertisement_id
Bidder_ID

2.Test Case Design

Delivery management

(IT22246332 – Hettiarachchi H.A.S.L)

Testing Function – Add shipping service provider

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Seller must be logged in to the system

Test Steps:

1. Input shipping provider details
2. Verify if a details with wrong validation
3. Submit the details
4. Verify if the details have been added to the system by checking the delivery person page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Fname-sandun Lname-lakshan Email- sandun1@gmail.com Number-0785573452 City-Gampaha Street-Negambo Road Driving License- A0000000	"Delivery person Added Successfully"	"Delivery person Added Successfully"	Pass	Confirm the delivery person data

Testing Function – Updating Shipping service provider

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Seller should be registered in the system.

Test Steps:

1. click “Edit” button in action field

2. Update the shipping provider’s details

3. Submit the updated details

4. Verify if the shipping provider’s details has been updated to the system by checking the my orders page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Number-0759546321 City-Colombo Street-high level Road	“Delivery person updated Successfully”	“Delivery person updated Successfully”	Pass	Update details of specific person

Testing Function – Deleting Sipping service provider

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Seller should be registered in the system.

Test Steps:

1. click “delete” button in action field

2. Confirm the deletion by clicking “Yes”

3. Verify if the shipping service provider has been deleted from the system by checking the my orders page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Clicking “Delete” button on the delivery person page	“Delivery person deleted Successfully”	“Delivery person deleted Successfully”	Pass	Deleting shipping provider from the system

Testing Function – Add delivery details

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Shipping provider must be logged in to the system

Test Steps:

1. Input delivery details
2. Verify if a details with wrong validation
3. Submit the details
4. Verify if the details have been added to the system by checking my orders page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Date – 2024/10/8 Time – 09.00 States – on the way	“Delivery states Added Successfully”	“Delivery states Added Successfully”	Pass	Confirm the delivery data

Testing Function – Updating delivery states

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Shipping service provider should be registered in the system.

Test Steps:

1. click “Edit” button in action field
2. Update the delivery states
3. Submit the updated details
4. Verify if the delivery states has been updated to the system by checking the my orders page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Date – 2024/10/10 Time – 12.00 States – late	“Delivery states updated Successfully”	“Delivery states updated Successfully”	Pass	Confirm the delivery data

Testing Function – Deleting delivery details

Test case designed by – IT22246332

Test Priority – High

Pre-condition – Shipping provider should be registered in the system.

Test Steps:

1. click “delete” button in action field
2. Confirm the deletion by clicking “Yes”
3. Verify if the delivery states has been deleted from the system by checking the my orders page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Clicking “Delete” button on the my orders page	“Delivery states deleted Successfully”	“Delivery states deleted Successfully”	Pass	Confirm the delivery data

i. Item Management system (IT22066770)

1.1 Testing Function – Add an Item

- Test case designed by – IT22066770
- Test Priority – High
- Pre-condition – Auction Manager must be logged in to the system.

Test Steps:

1. Input Item details
2. Verify if a Item with the same item code does not exist through form validation
3. Submit the details
4. Verify if the product has been added to the system by checking the Item List page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CI - 01	Item code: DP320, Item name: melissa art Unit price: $12000 Category: Arts,	“Item Added Successfully”	“Item Added Successfully”	Pass	Confirm the creation of the product data

1.2Testing Function – Updating a Item

- Test case designed by – IT22066770
- Test Priority – High
- Pre-condition – The Item should be Inserted in the system.

Test Steps:

1.Select Item and click “Edit” button

2. Update the relevant Item details
3. Submit the updated details
4. Verify if the Item has been updated to the system by checking the Item List page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	Item code: DP320, Item name: melissa art Unit price: $15000 Category: Arts	“Item Updated Successfully”	“Item Updated Successfully”	Pass	Confirm the Updation of the product data

1.3 Testing Function – Deleting a Item

 Test case designed by – IT22066770

 Test Priority – High

 Pre-condition – The Item should be registered in the system.

Test Steps:

1. Select Item and click “Delete” button
2. Confirm the deletion by clicking “Yes” in the popup
3. Verify if the Item has been deleted from the system by checking the Item List page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	Clicking “Delete” button on the Item page	“Item Deleted Successfully”	“Item Deleted Successfully”	Pass	Deleting a Item from the system

1.4 Testing Function – Search Item

 Test case designed by – IT22066770

 Test Priority – Medium

 Pre-condition – At least a few Items should be registered in the system.

Test Steps:

1. Type a Item name

2. Click search

3. View the results

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	“Melissa Art”	All the Items that have the word “Melissa” in their name”	All the Items that have the word “Melissa” in their name”	Pass	Confirming the report search function is operating without errors.

ii. Auction Management system (IT22066770)

1.2 Testing Function – Add an Auction

 Test case designed by – IT22066770

 Test Priority – High

 Pre-condition – Auction Manager must be logged in to the system.

Test Steps:

1. Input Auction details
2. Verify if a Auction with the same auction code does not exist through form validation
3. Submit the details
4. Verify if the Auction has been added to the system by checking the Auctions page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CA - 01	auction code: DP320, auction title: melissa art auction Category: Arts, Date:2024/10/5	"Auction Added Successfully"	"Auction Added Successfully"	Pass	Confirm the creation of the Auction data

1.2 Testing Function – Updating an Auction

 Test case designed by – IT22066770

 Test Priority – High

 Pre-condition – The Auction should be Inserted in the system.

Test Steps:

1. Select Item and click "Edit" button
2. Update the relevant Auction details
3. Submit the updated details
4. Verify if the Auction has been updated to the system by checking the Auction page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	auction code: DP320, auction title: melissa art auction Category: Arts, Date:2024/10/8	“Auction Updated Successfully”	“Auction Updated Successfully”	Pass	Confirm the Updating of the Auction data

1.4Testing Function – Deleting a Auction

 Test case designed by – IT22066770

 Test Priority – High

 Pre-condition – The Auction should be registered in the system.

Test Steps:

1. Select Auction and click “Delete” button
2. Confirm the deletion by clicking “Yes” in the popup
3. Verify if the Auction has been deleted from the system by checking the Auction page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	Clicking “Delete” button on the Auction page	“Auction Deleted Successfully”	“Auction Deleted Successfully”	Pass	Deleting a Auction from the system

1.4 Testing Function – Search Auction

 Test case designed by – IT22066770

 Test Priority – Medium

 Pre-condition – At least a few Auctions should be registered in the system.

Test Steps:

1. Type an Auction name

2. Click search

3. View the results

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UI - 01	“Melissa Art auction”	All the Auction that have the word “Melissa” in their name”	All the Auction that have the word “Melissa” in their name”	Pass	Confirming the report search function is operating without errors.

Bidding mechanism
(IT22606174 Samaranayake W.M.R.L)

1.1 Testing Function – Display highest bid for an item

- Test case designed by – IT22606174
- Test Priority – High
- Pre-condition – An auction is active, and there are existing bids.

Test Steps:

1. Navigate to the auction page of an item.
2. View the highest bid displayed.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
HI - 01	Item ID: 12345	"The system displays the highest bid.	"The system shows the highest bid (e.g. $500)"	Pass	The system correctly retrieved and displayed the highest bid for an item that has ongoing bids.

1.2Testing Function – Receive confirmation after successful bid

- Test case designed by – IT22606174
- Test Priority – low
- Pre-condition – A bid should be successfully placed.

Test Steps:

1.Navigate to the auction page of an item.
2.Place a valid bid on an item.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
NT - 01	Item ID: 12345 Bid amount: $550	“Bid places successfully”	“Your bid was successfully placed”	Pass	The system successfully processed the bid and showed the “successful” notification.

1.3 Testing Function – Prevent outbidding oneself

- Test case designed by – IT22606174
- Test Priority – High
- Pre-condition – The bidder is the current highest bidder on the item.

Test Steps:

1. Navigate to the auction page.
2. Enter a bid that is higher than the current bid but still outbids the bidder themselves
3. Submit the bid.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
OB - 01	Item ID: 12345 New bid: $600	The system should reject the bid and display a message	“You cannot outbid yourself”	Pass	The system correctly prevented the user from outbidding themselves and showed an appropriate error message

1.4 Testing Function – Notify the bidder upon winning the auction

- Test case designed by – IT22606174
- Test Priority – High
- Pre-condition – The auction has ended, and the bidder has the highest bid.

Test Steps:

1. Navigate to an auction item (Item ID: 12345)
2. Ensure the auction countdown timer reaches 00:00:00
3. Ensure the bidder's bid is the highest bid when the auction ends.
4. Verify that a notification is sent to the bidder upon winning the auction.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
NT - 02	Item ID: 12345 Place highest bid: $300	The bidder receives a notification	"Congratulations! You have won the auction for Item 12345."	Pass	Notify the bidder at auction end; failure if no notification is received.

Advertising Handling System

(IT22224170 – Sanjana K.G.T.S)

Testing Function – Create an Advertisement

- Test case designed by = IT22224170
- Test Priority = High
- Pre-Condition = Advertisement Handler must be logged to the system

Test Steps:

1. Input the Advertisement Details.
2. Verify if an advertisement in the right date through form validation
3. Submit the details
4. Verify if the advertisement has been added to the system by checking the advertisement details page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CA - 01	Image: URL. Title: Jewelry, Collectables, Art. Date:2024/09/28 Description: Art Auction SWRD Bandaranaike National Memorial Foundation Bauddhaloka Mawatha, Colombo 07.	"Advertisement Added Successfully"	"Advertisement Added Successfully"	Pass	Confirm the creation of the Advertisements data

Testing Function – Updating an Advertisement

- Test case designed by = IT22224170
- Test Priority = High
- Pre-Condition = Advertisement should be added to the system

Test Steps:

1. Select the Advertisement and click "Update" button.
2. Update the advertisement details.
3. Submit the updated advertisement details.
4. Verify if the advertisement has been updated to the system by checking the advertisement details page.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UA - 01	Image: URL. Title: Jewelry, Date:2024/09/30 Description: Jewelry Auction Kandy City Centre · 5, Dalada Veediya, Kandy · Sri Lanka	"Advertisement Updated Successfully"	"Advertisement Updated Successfully"	Pass	Updating details of a specific advertisemen t

Testing Function – Deleting an Advertisement

- Test case designed by = IT22224170
- Test Priority = High
- Pre-Condition = Advertisement should be added to the system

Test Steps:

1. Select the Advertisement and click "Delete" button.
2. Confirm the deletion by clicking "Yes" in the popup.
3. Verify if the advertisement has been deleted from the system by checking the advertisement details page.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
DA - 01	Clicking "Delete" button on the advertisement details page	"Advertisement Deleted Successfully"	"Advertisement Deleted Successfully"	Pass	Deleting an advertisemen t from the system

Testing Function – Generate Advertisement Report

- Test case designed by = IT22224170
- Test Priority =Medium
- Pre-Condition = Advertisement Handler must be logged to the system

Test Steps:

1. Generate the report.
2. Download the pdf file.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
RA - 01	Click "Generate Report" Button.	PDF file with the records downloading	PDF file with the records downloading	Pass	Confirm the report generation function is functioning

Testing Function – Search Advertisement

- Test case designed by = IT22224170
- Test Priority =Medium
- Pre-Condition = At least a few Advertisements should be added into the system

Test Steps:

1. Go to the advertisement details page.
2. Type the keyword (Title/Date) on the search.
3. Verify all the advertisements that include that keyword are fetched

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
SA - 01	Existing order records.	Relevant search results are showing.	All the advertisements records are fetched and displayed on the page	Pass	When any keyword is entered the orders will be filtered and show the search results.

User Registration management

(IT22215192 - Viduranga S P S)

Testing Function - Add User

Test case designed by - IT22215192

Test Priority - High

Pre-condition - User must be fill in the registration form

Test Steps:

1. Input Bidder personal details
2. Verify if a details with wrong validation
3. Submit the details
4. Verify if the details have been added to the system by checking the Bidder profile page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	First name - Sahan Last name - Viduranga Email - sahan@gmail.com Contact information - 0702340694 Gender - Male Date of birth - 08/12/2002 Address - 12 / Gampaha Username - Sahan Password – sahan123	"Bidder Added Successfully"	"Bidder Added Successfully"	Pass	Confirm the bidder details

Testing Function - Updating Bidder details

Test case designed by - IT22215192

Test Priority - High

Pre-condition - Bidder should be registered in the system.

Test Steps:

1. click “Edit profile” button in bidder profile page
2. Update the bidder’s details
3. Submit the updated details
4. Verify if the bidder details have been updated to the system by checking the profile page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Email – sahan7sv@gmail.com Contact information - 0702078954 Address - 15 / Colombo	“Bidder data updated Successfully”	“Bidder data updated Successfully”	Pass	Update details of specific person

Testing Function – Deleting Bidder

Test case designed by – IT22215192

Test Priority – High

Pre-condition – Bidder should be registered in the system.

Test Steps:

1. click “delete” button in registration manager admin part
2. Confirm the deletion by clicking “Yes”
3. Verify if the bidder has been deleted from the system by checking the manage bidder page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
	Clicking “Delete” button on the manage bidder page	“Bidder deleted Successfully”	“Bidder deleted Successfully”	Pass	Deleting bidder from the system

Finance Handling
(IT22238580 – Pihillegedara S.N.M)

1.1 Testing Function – Create -Add Salary

- Test case designed by – IT22238580
- Test Priority – High Pre-condition – Financial Manager must be logged in to the system.

Test Steps:

1. Input the basic salary, bonuses, OT hours , OT rates according to the employee job roles
2. Verify if the employee is registered to the system and the EPF & ETF rates are correctly deducted.
3. Submit the details by clicking Add Salary button.
4. Verify if the salary has been added to the employee salary table in the system by checking the Employee management page

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CS - 01	Basic Salary-100000.00 Bonus -3000.00 OT Hours-8 OT rate-300.00	“Total Salary Added Successfully”	“Total Salary Added Successfully”	Pass	Confirm the calculation of the total salary

1.2 Testing Function – Updating the Salary Details

- Test case designed by – IT22238580
- Test Priority – High Pre-condition – The Employee should be registered in the system.

  Test Steps:
  1. Select the employee and click “Update Salary” button
  2. Update the relevant salary details
  3. Submit the updated details by clicking “Add salary” button
  4. Verify if the salary has been updated to the system by checking the salary table.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
US - 01	Basic Salary-100000.00 Bonus-4000.00 OT hours-8 OT Rate -300.00	“Salary Updated Successfully”	“Salary Updated Successfully”	Pass	Updating details of a salary record

1.3 Testing Function – Deleting a Salary Record

- Test case designed by – IT22238580
- Test Priority – High Pre-condition – The calculated total salary should be displayed on the Employee salary table

Test Steps:

1. Select the Employee from the search bar
2. click "Delete" button
3. Verify if the employee total salary has been deleted from the system by checking the salary table in employee management page

1.4 Testing Function – Creating a Cash Record

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
DS - 01	Clicking "Delete" button on the salary table.	"Salary deleted Successfully"	"Salary deleted Successfully"	Pass	Deleting a salary record from the system

- Test case designed by – IT22238580
- Test Priority – High Pre-condition – Financial Manager must be logged in to the system

Test Steps:

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CC - 02	Cash Type- “Income” Date – 2024.09.29 Description“Capital” Amount-“500000”	“Cash Record Added Successfully”	“Cash Record Added Successfully”	Pass	Confirm the creation of th cash record

1.5 Testing Function – Updating a Cash Record

- Test case designed by – IT22238580
- Test Priority– High Pre-condition – The Cash Record should be added to the system.

Test Steps:

1. Select cash record and click “Edit” button
2. Update the relevant cash record details
3. Submit the updated details
4. Verify if the cash record has been updated to the system by checking the Income-Expenses table

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UC - 02	Cash Type- “Income” Date – 2024.09.29 Description“Capital” Amount-“700000”	“Cash Record Updated Successfully”	“Cash Record Updated Successfully”	Pass	Updating details of a specific salary record

1. Select the cash type whether that is a income, expense or a peticash
2. Input the date , Description and Amount.
3. Select the “Add Cash” button.
4. Verify if the cash record has been added to the system by checking the Income-Expenses table.

-

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
DC - 02	Clicking“Delete” button on the Table	“Cash Record deleted Successfully”	“Cash Record deleted Successfully”	Pass	Deleting a cash record

1.7 Testing Function – Generate a Monthly balance Report

- Test case designed by – IT22238580
- Test Priority – Medium Pre-condition – Financial Manager must be logged in to the system.

  Test Steps:
  1. Select a time period to generate the cash record.
  2. Confirm the time period
  3. Generate the report
  4. Download the PDF file

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
RC - 01	Time period: Last month Click “Generate report” button	PDF file with the records downloading	PDF file with the records downloading	Pass	Confirm the report generation function i functioning

1.6 Testing Function – Deleting a Cash Record

- Test case designed by –IT22238580
- Test Priority – High
- Pre-condition – The Cash Record should be added to the system

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
SS - 01	“nimal”	All the records that have the word “nimal” in their name	All the records that have the word “Adaptor” in their name	Pass	Confirming the report search function is operating without errors

Test Steps:

1. Select the cash record
2. Click “Delete” button.
3. Verify if the cash record has been deleted from the system by checking the table.

1.8 Testing Function – Search Record

- Test case designed by – IT22238580
- Test Priority – Medium Pre-condition – At least a few Employees should be registered in the system.

Test Steps:

1. Type an employee name or email
2. Click search

View the results

Payment Gateway Integration

(IT22091970 (Chandraguptha H.A.T))

Test Case 1: Processing Payments for Shipping Services

- **Test case designed by** – IT22091970 (Chandraguptha H.A.T)
- **Test Priority** – High
- **Pre-condition** – Buyer has selected an item for purchase and shipping provider.

Test Steps:

1. Ensure the buyer has valid payment credentials.

2.Initiate the payment through the gateway.

3. Verify payment completion via the payment gateway API.

4. Check if the payment is logged in the Transactions Table.

5. Confirm shipping provider and seller are notified of payment completion.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
PG001	**Buyer Payment Info:** Card Number: 4111111111111111 Expiration Date: 12/2025 CVV: 123 Amount: $100	Payment processed successfully	Payment processed successfully	Pass	Ensure the payment gateway processes valid payment

Test Case 2: Verifying Payment Status During Delivery

- **Test case designed by** – IT22091970 (Chandraguptha H.A.T)
- **Test Priority** – Medium
- **Pre-condition** – Payment has been initiated for an item

Test Steps:

1. Trigger the shipping process once payment is verified.
2. Update delivery status in the system (e.g., "In Transit").
3. Verify payment status is updated in the gateway's log.
4. Confirm that no payment is released until the delivery status is "Delivered."

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
PG101	**Order Info:** Order ID: 12345 Payment Status: Paid Delivery Status: In Transit	Payment status remains "Paid" in the gateway log	Payment status remains "Paid" in the gateway log	Pass	Confirm payment status remains unchanged during transit

Test Case 3: Releasing Payments to Sellers and Shipping Providers

- **Test case designed by** – IT22091970 (Chandraguptha H.A.T)
- **Test Priority** – High
- **Pre-condition** – Item has been delivered.

Test Steps:

1. Check if the delivery status is updated to "Delivered."
2. Automatically trigger payment release to the seller and shipping provider.
3. Verify if payment receipts are generated for both the seller and shipping provider.
4. Log the payment details in the system.

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
PG20	**Order Info:** Order ID: 98765 Delivery Status: Delivered	Payment released to seller and shipping provider	Payment released to seller and shipping provider	Pass	Verify payment release upon delivery completion

<u>Auction Event Handling (IT22235138 – Thalangama A. I)</u>

1.1 Testing Function – Create - Assign a Seat

- Test case designed by – IT22235138
- Test Priority – High
- Pre-condition - Seating Manager must be logged in to the system.

Test Steps:

1. Input the seat Id, bidder Id, bidder name, email & location.
2. Verify if there are no assigned seat Id in same location through form validation.
3. Submit the details by clicking submit button.
4. Verify if the seat Id and bidder details has been added to seat list table in the system by checking the seat table

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CS - 01	SeatId – A1 BidderId – 22 Name – Anupama Email – anu@gmail.com Location - BMICH	“Seat assign Successfully”	“Seat assign Successfully”	Pass	Confirm the seat assignment and inform bidder via email

1.2 Testing Function – Updating the Seat Assignment

- Test case designed by – IT22235138
- Test Priority – High
- Pre-condition – The seat should be assigned for a bidder in the system.

Test Steps:

1. Select the seat Id and click “Update” button
2. Update the relevant seat Id or other details
3. Submit the updated details by clicking “Update Seat” button
4. Verify if the details have been updated to the system by checking the seat table and seating plan

1.3 Testing Function – Deleting an Assigned Seat

- Test case designed by – IT22235138
- Test Priority – High
- Pre-condition – The seat should be assigned for a bidder in the system.

Test Steps:

1. Select the seat of seat table
2. click “Delete” button
3. Verify if the assigned seat has been deleted from the system by checking the seat table

1.4Testing Function – Add Rows and Columns to Seating Plan

- Test case designed by – IT22235138
- Test Priority – Medium
- Pre-condition – Seating Manager must be logged in to the system

Test Steps:

1. Select the seating plan of a specific location
2. Click add row or add column button
3. Verify if new row or new column has been deleted from the seating plan

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
CC - 02	Clicking the “Add Row” button or “Add Column” button of seating plan	“Add new row or column Successfully”	“Add new row or column Successfully”	Pass	Confirm the adding new rows and columns to the seating plan

1.5 Testing Function – Delete Rows And Columns of Seating Plan

- Test case designed by – IT22235138
- Test Priority– Medium
- Pre-condition – Seating Manager must be logged in to the system.

Test Steps:

1. Select the seating plan of a specific location
2. Click “delete row” or “delete column” button
3. Verify if last row or column has been deleted from the seating plan

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
UC - 02	Clicking the "Delete Row" button or "Delete Column" button of seating plan	"Delete row or column Successfully"	"Delete row or column Successfully"	Pass	Confirm the deleting rows and columns the seating plan

1.7Testing Function – Generate an Auction Report

- Test case designed by – IT22235138
- Test Priority – Medium
- Pre-condition – Seating Manager must be logged in to the system.

Test Steps:

1. Select the location to generate the report.
2. Generate the report
3. Download the PDF file

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
RC - 01	Click "Generate report" button	PDF file with the details downloading	PDF file with the details downloading	Pass	Confirm the report generation function is functioning

1.8 Testing Function – Search Seat

- Test case designed by – IT22235138
- Test Priority – Medium
- Pre-condition – At least a few seats should be assigned in the system.

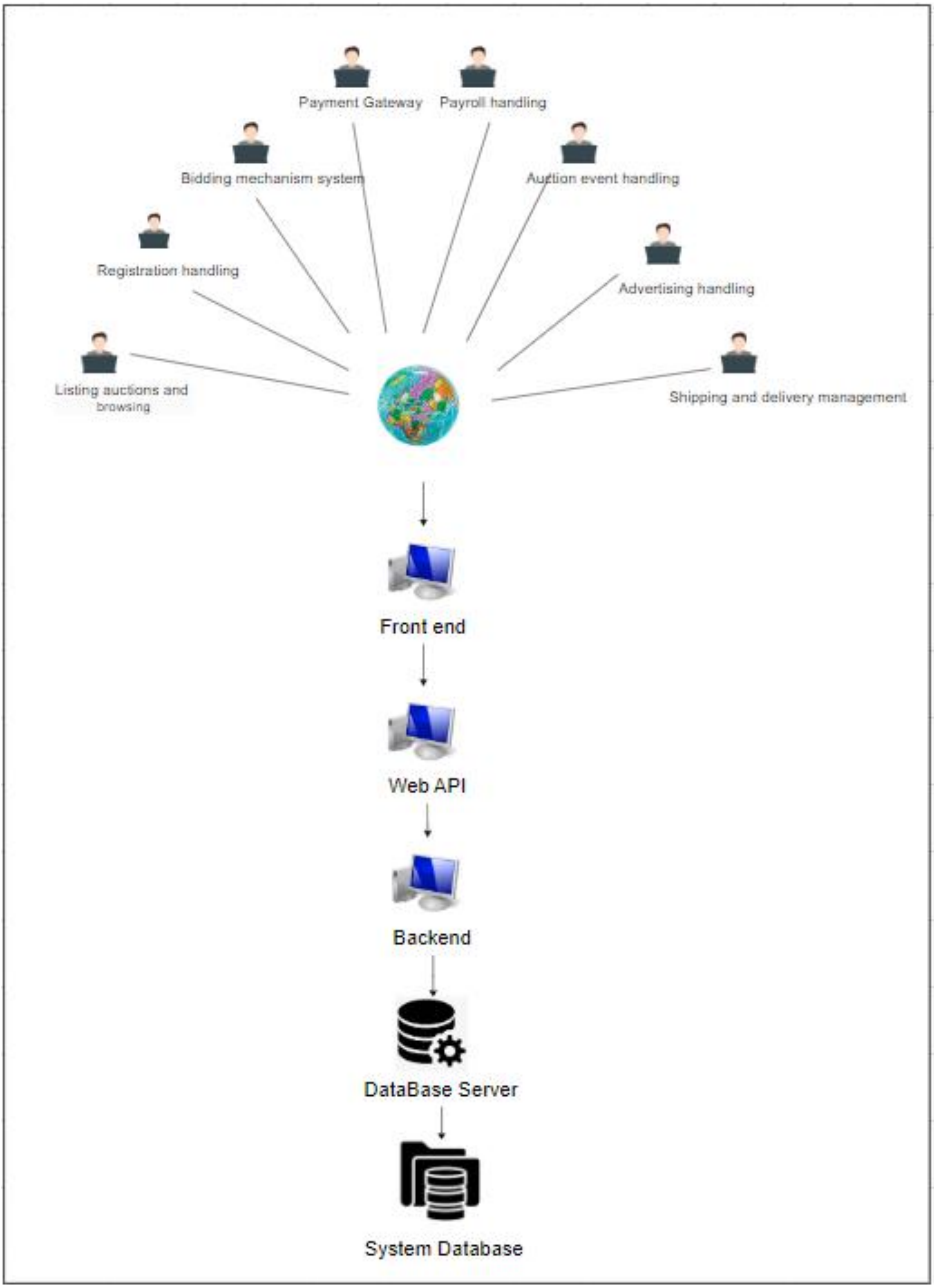
Test Steps:

1. Type a seat Id
2. Click search
3. View the results
4. Click filter and select location

View the results

Test ID	Test inputs	Expected output	Actual output	Result (Pass / Fail)	Description
SS - 01	"A1"	All the records that have the seat Id "A1" in different locations	All the records that have the seat Id "A1" in different locations	Pass	Confirming the seat search function is operating without errors

5.System Diagram



Work Progression & Technical Specifications

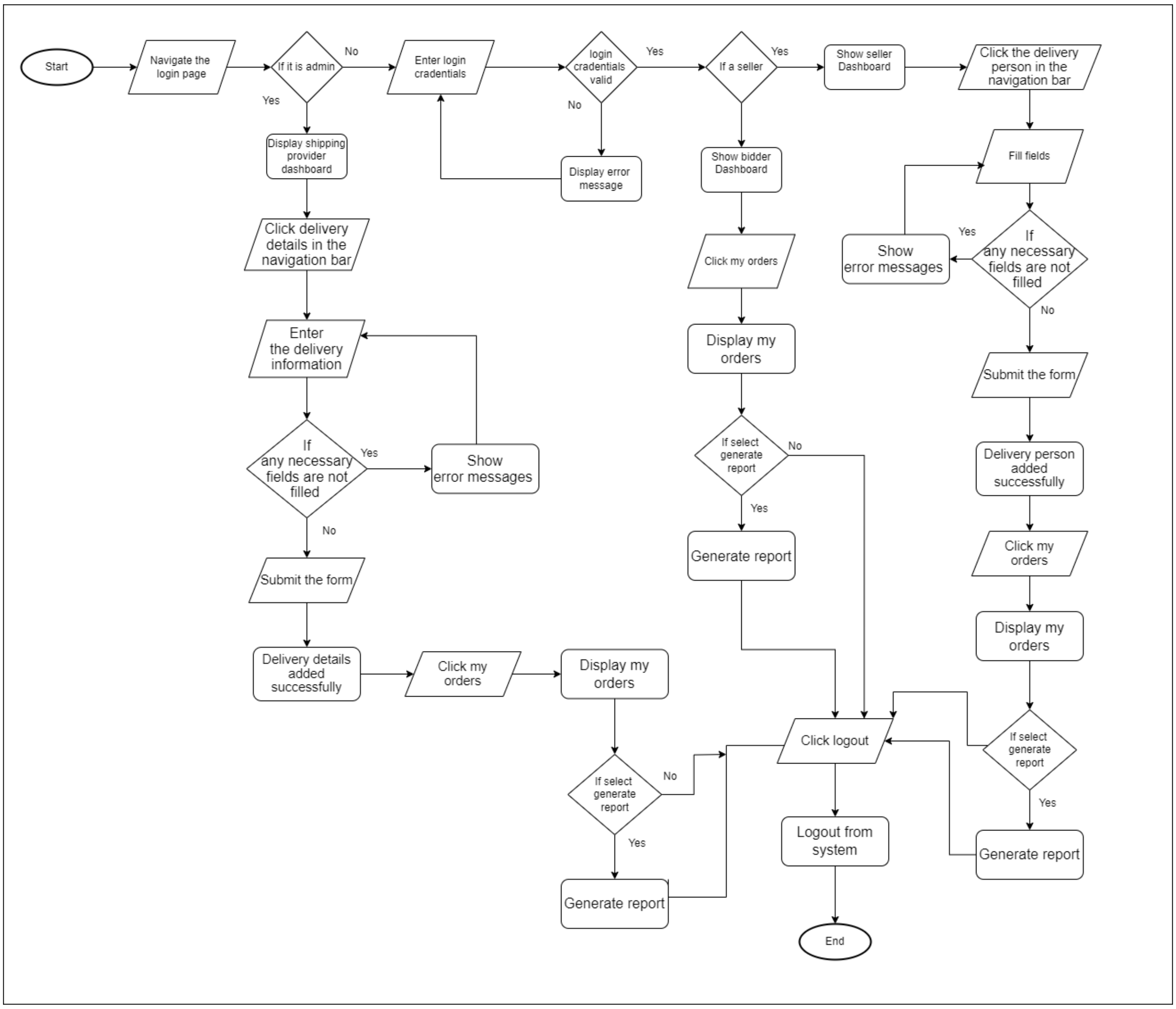
Delivery management (IT22246332 -Hettiarachchi H.A.S.L)

The Delivery Management role is a crucial component of the overall delivery process, managing the flow of goods from the point of dispatch to the destination. It guarantees the efficient coordination of several crucial elements, such as order dispatch, delivery scheduling, real-time tracking, and performance reporting, under the supervision of the delivery manager
This component entails the meticulous upkeep of a sizable database including comprehensive delivery data. It guarantees accurate real-time logging and updating of delivery facts, including customer address, item weight, and delivery status.
This component entails the meticulous upkeep of a sizable database including comprehensive delivery data. It guarantees accurate real-time logging and updating of delivery facts, including customer address, item weight, and delivery status.
From shipment to final delivery confirmation, the system keeps a tight eye on every delivery's status. In order to give real-time updates on delivery status, driver location, and estimated time of arrival (ETA), it combines GPS-based tracking systems. Every delivery is tracked thanks to the dynamic monitoring tool, which also notifies customers automatically when milestones are met. Based on delivery records, this mechanism creates detailed reports that provide important insights into delivery performance, average delivery times, route efficiency, and any problems or delays experienced. The reports are crucial instruments for optimizing delivery effectiveness, monitoring operational efficiency, and facilitating data-driven decision-making.

Work Progression

Completed Work	Work in Progress	Work to be Completed
Implementing CRUD operations with validations for sold items, delivery persons and delivery states database collections Implementing a search function to search delivery items and delivery persons by the name Implementing a function to generate reports for my orders Records Linking sold items, delivery persons and my orders with other functions	Improving system delivery security to show each delivery details for relevant sellers and bidders.	Implement a feature that allows customers to cancel or reschedule deliveries through the system before the dispatch stage. Display a real-time dashboard for the delivery manager to monitor ongoing deliveries and track any delays.

Flow chart



Start
Navigate the login page
If it is admin
No
Yes
Enter login credentials
login cradentials valid
Yes
No
Display error message
If a seller
Yes
Show seller Dashboard
Click the delivery person in the navigation bar
Display shipping provider dashboard
Show bidder Dashboard
Fill fields
Click delivery details in the navigation bar
Click my orders
Show error messages
If any necessary fields are not filled
Yes
No
Enter the delivery information
Display my orders
Submit the form
If any necessary fields are not filled
Yes
Show error messages
If select generate report
No
Yes
Delivery person added successfully
No
Generate report
Click my orders
Submit the form
Display my orders
Delivery details added successfully
Click my orders
Display my orders
Click logout
If select generate report
If select generate report
No
Yes
Logout from system
Yes
Generate report
Generate report
End

Algorithm:

Step 1: Start

Step 2: Access Bidmaster web application

Step 3: Login to the system
  a. If entered credentials are incorrect
    - Display "Entered username and password don’t match"
    - Return to Step 3

  b. If correct credentials are entered

      i. If the user is a seller
        - Proceed to Step 4
      ii. If the user is a shipping provider
        - Proceed to Step 8

Step 4: Show the Dashboard once you've signed in as a seller.

  a. If select generate pdf link
    -Generate report as a pdf

Step 5: Click the option in the navigation bar
  a. If "Add Delivery Person" is selected
    - The "Add Delivery Person" form will appear.

Step 6: Complete the "Add Delivery Person" form with your information.

  a. If any necessary fields are not filled in or if incorrect data is entered
    - Show error messages in the corresponding fields.
    - Return to Step 6

  b. If all fields correctly filled
    - Proceed to Step 7

Step 7: Submit the form
  a. Should the submission be accepted
    - Display "Delivery person added successfully"
    - Return to the dashboard

  b. If there is a problem with the submission
    - Display the error message
    - Return to Step 6

```
Step 8: After logging in as a shipping provider, display the Dashboard.

Step 9: Click the option in the navigation bar
  a. If "Enter Delivery Details" is selected
    - Display the "Enter Delivery Details" form

Step 10: Enter the delivery information, such as:
    - Delivery Date
    - Delivery Time
    - Delivery Status

  a. If any necessary fields are not filled
    - Display error messages
    - Return to Step 10

  b. If all fields are correctly filled
    - Proceed to Step 11

Step 11: Submit the  form
  a. If the successful submission
    - Display "Delivery details entered successfully"
    - Return to the dashboard

  b. If there is a problem with the submission
    - Display the error message
    - Return to Step 10


Step 12: Logout of the system (if needed)

End
```

Pseudocode:

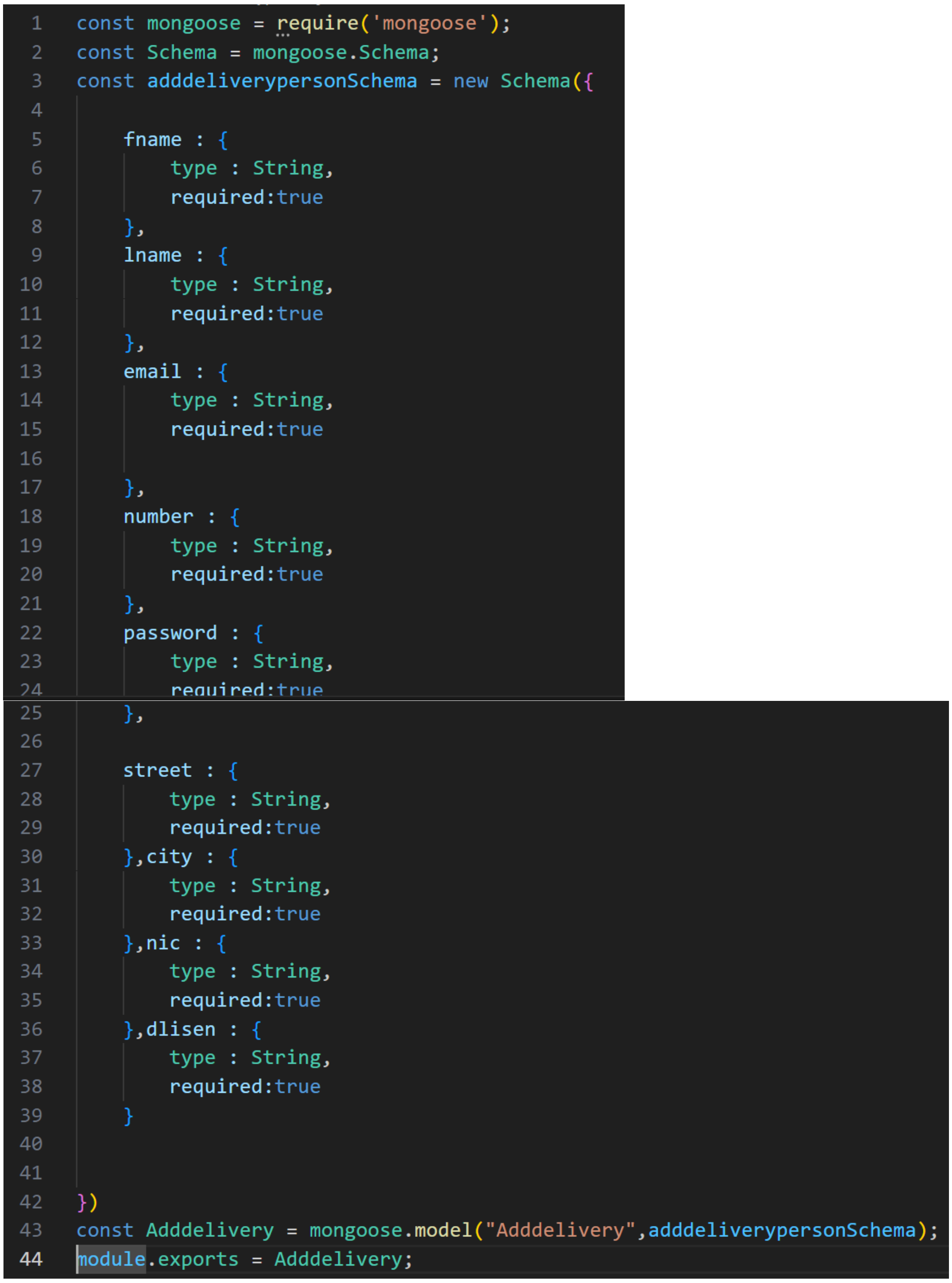
```
BEGIN

Step 1: Start the application

Step 2: Access Bidmaster web application

Step 3: Login to the system
    IF credentials are incorrect THEN
        DISPLAY "Entered username and password don’t match"
        RETURN to Step 3
    ELSE
        IF user is a seller THEN
            GO TO Step 4
        ELSE IF user is a shipping provider THEN
            GO TO Step 8
        END IF
    END IF

Step 4: Display seller's Dashboard
    IF "Generate PDF" link is selected THEN
        GENERATE report as PDF
    END IF

Step 5: Click option from navigation bar
    IF "Add Delivery Person" is selected THEN
        DISPLAY "Add Delivery Person" form
    END IF
```

```
Step 6: Complete the "Add Delivery Person" form
    IF any required fields are empty or incorrect THEN
        DISPLAY error messages for corresponding fields
        RETURN to Step 6
    ELSE
        GO TO Step 7
    END IF

Step 7: Submit the form
    IF submission is successful THEN
        DISPLAY "Delivery person added successfully"
        RETURN to Dashboard
    ELSE
        DISPLAY submission error message
        RETURN to Step 6
    END IF

Step 8: Display shipping provider's Dashboard

Step 9: Click option from navigation bar
    IF "Enter Delivery Details" is selected THEN
        DISPLAY "Enter Delivery Details" form
    END IF

Step 10: Enter delivery information (Date, Time, Status)
    IF any required fields are empty THEN
        DISPLAY error messages for corresponding fields
        RETURN to Step 10
    ELSE
        GO TO Step 11
```

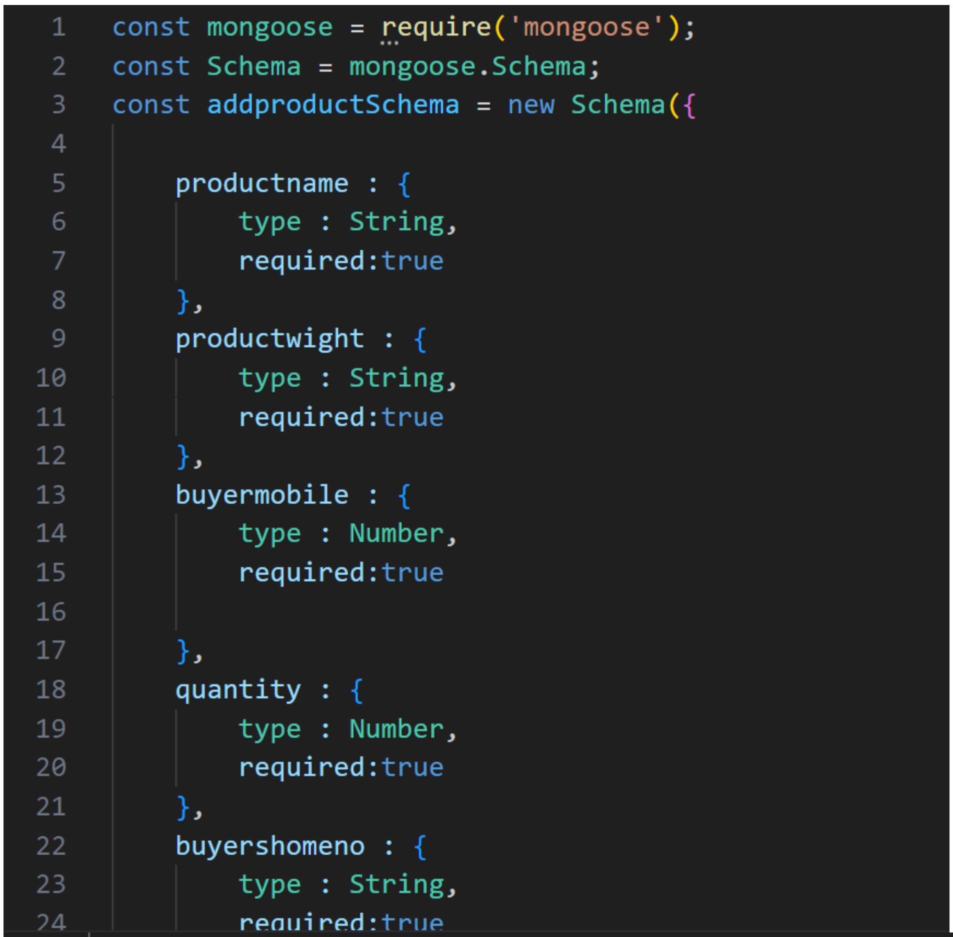
```
END IF


Step 11: Submit the form
    IF submission is successful THEN
        DISPLAY "Delivery details entered successfully"
        RETURN to Dashboard
    ELSE
        DISPLAY submission error message
        RETURN to Step 10
    END IF


Step 12: Logout if necessary


END
```

Database Schemas:

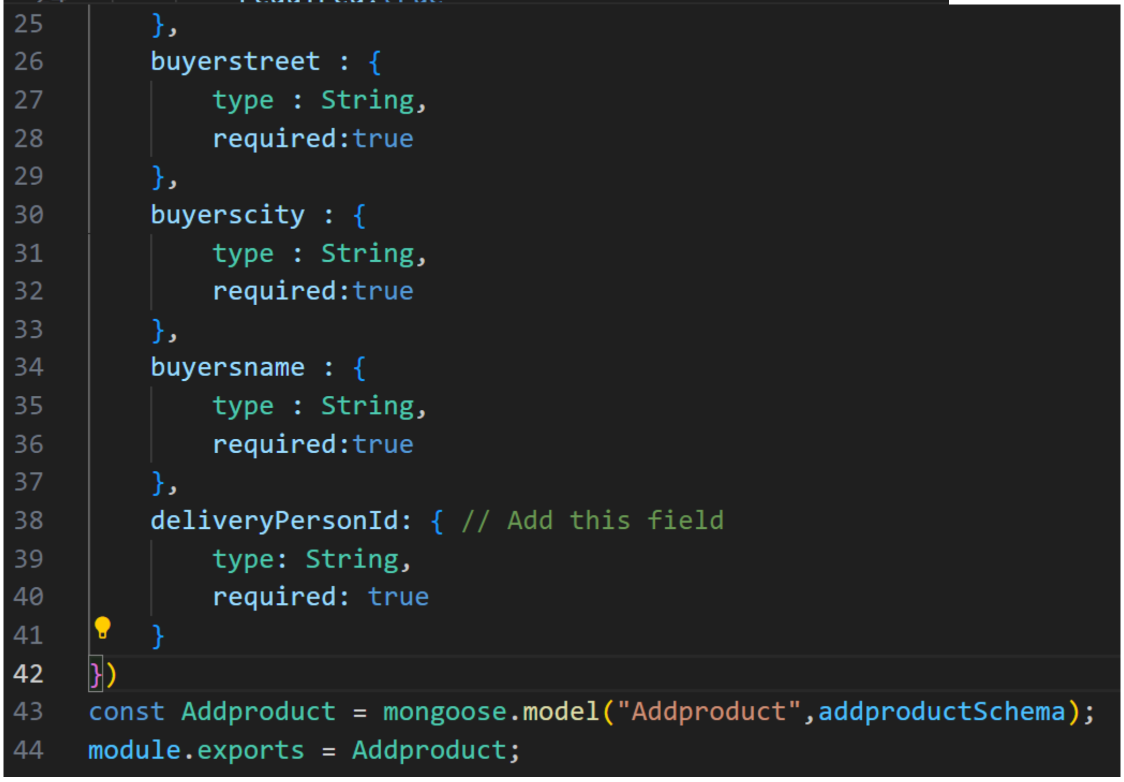
Add delivery person model



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const adddeliverypersonSchema = new Schema({

    fname : {
        type : String,
        required:true
    },
    lname : {
        type : String,
        required:true
    },
    email : {
        type : String,
        required:true

    },
    number : {
        type : String,
        required:true
    },
    password : {
        type : String,
        required:true
    },

    street : {
        type : String,
        required:true
    },city : {
        type : String,
        required:true
    },nic : {
        type : String,
        required:true
    },dlisen : {
        type : String,
        required:true
    }


})
const Adddelivery = mongoose.model("Adddelivery",adddeliverypersonSchema);
module.exports = Adddelivery;
```

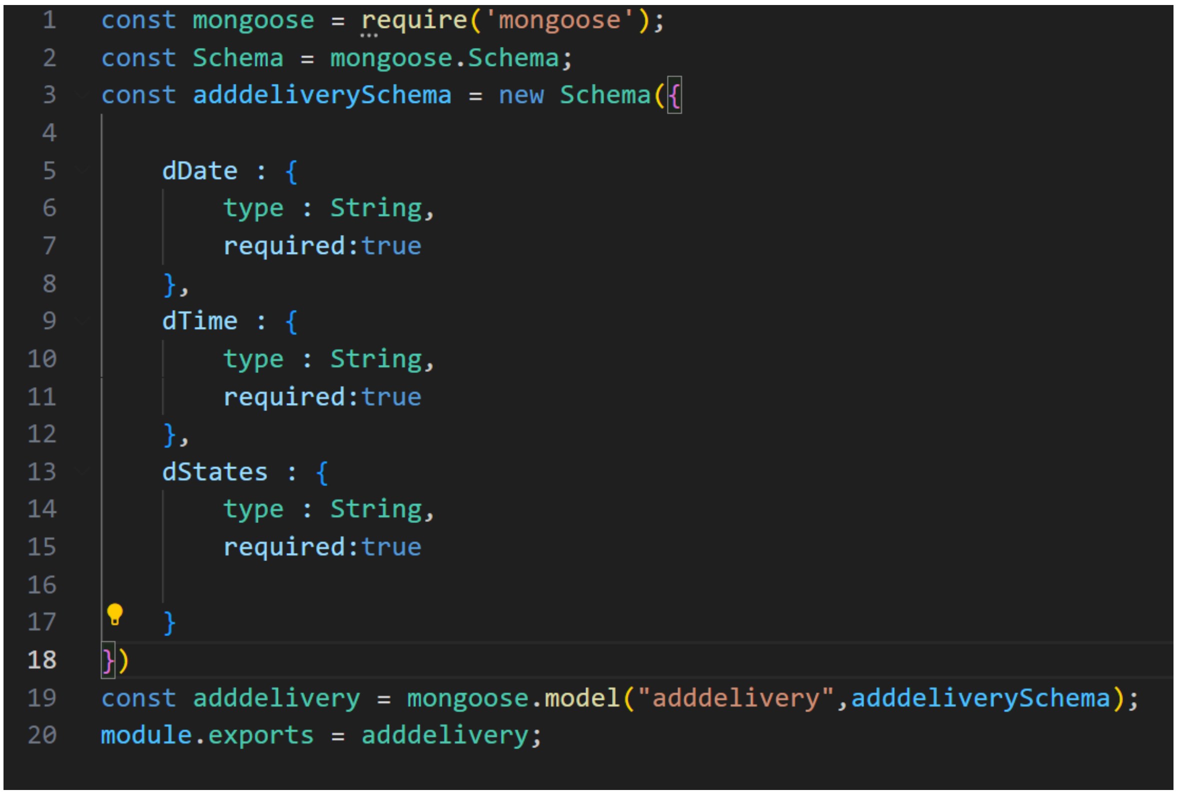


```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const addproductSchema = new Schema({

    productname : {
        type : String,
        required:true
    },
    productwight : {
        type : String,
        required:true
    },
    buyermobile : {
        type : Number,
        required:true

    },
    quantity : {
        type : Number,
        required:true
    },
    buyershomeno : {
        type : String,
        required:true
```



```
    },
    buyerstreet : {
        type : String,
        required:true
    },
    buyerscity : {
        type : String,
        required:true
    },
    buyersname : {
        type : String,
        required:true
    },
    deliveryPersonId: { // Add this field
        type: String,
        required: true
    }
})
const Addproduct = mongoose.model("Addproduct",addproductSchema);
module.exports = Addproduct;
```

Add Delivery Model



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const adddeliverySchema = new Schema({

    dDate : {
        type : String,
        required:true
    },
    dTime : {
        type : String,
        required:true
    },
    dStates : {
        type : String,
        required:true

    }
})
const adddelivery = mongoose.model("adddelivery",adddeliverySchema);
module.exports = adddelivery;
```

Seller Management (IT22066770)

An online auction management system's seller management feature is a tool that gives sellers the ability to handle a variety of duties, from registration to auction closing. Below is a summary of its main features:

Registration: By entering their name, contact information, and preferred method of payment, sellers can register for an account on the platform. To confirm the legitimacy of the seller, the registration process may involve identification or email verification.

Creation of Accounts: Following registration, sellers have the option to construct comprehensive profiles that contain personal or company data, seller ratings, previous transaction history, and other pertinent credentials. This profile increases seller visibility and fosters confidence with prospective customers.

Listing of Items: Vendors can place goods up for auction by including thorough descriptions, high-quality photos, and specifics about the goods, like its category, beginning price, and condition. In order to properly display their wares and draw in more bidders, sellers are able to post multiple photographs.

Auction Management: Sellers determine the terms of each auction, such as the beginning and ending dates, the minimum reserve amount, and the increments in bids. They can select from a variety of auction kinds (such as reserve auctions, buy-now options, or normal auctions) and change the parameters if needed while the sale is in progress.

Inventory Management: Using a dashboard, sellers can adjust item details, keep an eye on active bids, check the status of their auctions, and even relist unsold items. If a vendor is selling more than one item, this tool also lets them monitor stock levels.

Bid Monitoring: In real time, sellers may keep tabs on the bidding activity on their merchandise. When a bid is made, they are alerted, which enables them to monitor the status of the auction and address any problems.

Buyer Interaction: The system offers communication features that enable sellers to reply to questions from buyers or engage in direct negotiations with possible sellers. This could be responding to enquiries about the product, going over delivery choices, or outlining the conditions of payment.

Transportation & delivery: Following the conclusion of an auction, sellers are in charge of handling delivery and shipment. The platform might let merchants select their own shipping options, or it might provide integrated shipping solutions. Additionally, sellers can provide purchasers with tracking information to guarantee open and honest communication.

Payment Processing: The platform's integrated secure payment channels are used by sellers to collect payments. These payment options guarantee prompt transactions between buyers and sellers, and the platform frequently retains money in escrow until the deal is successfully completed.

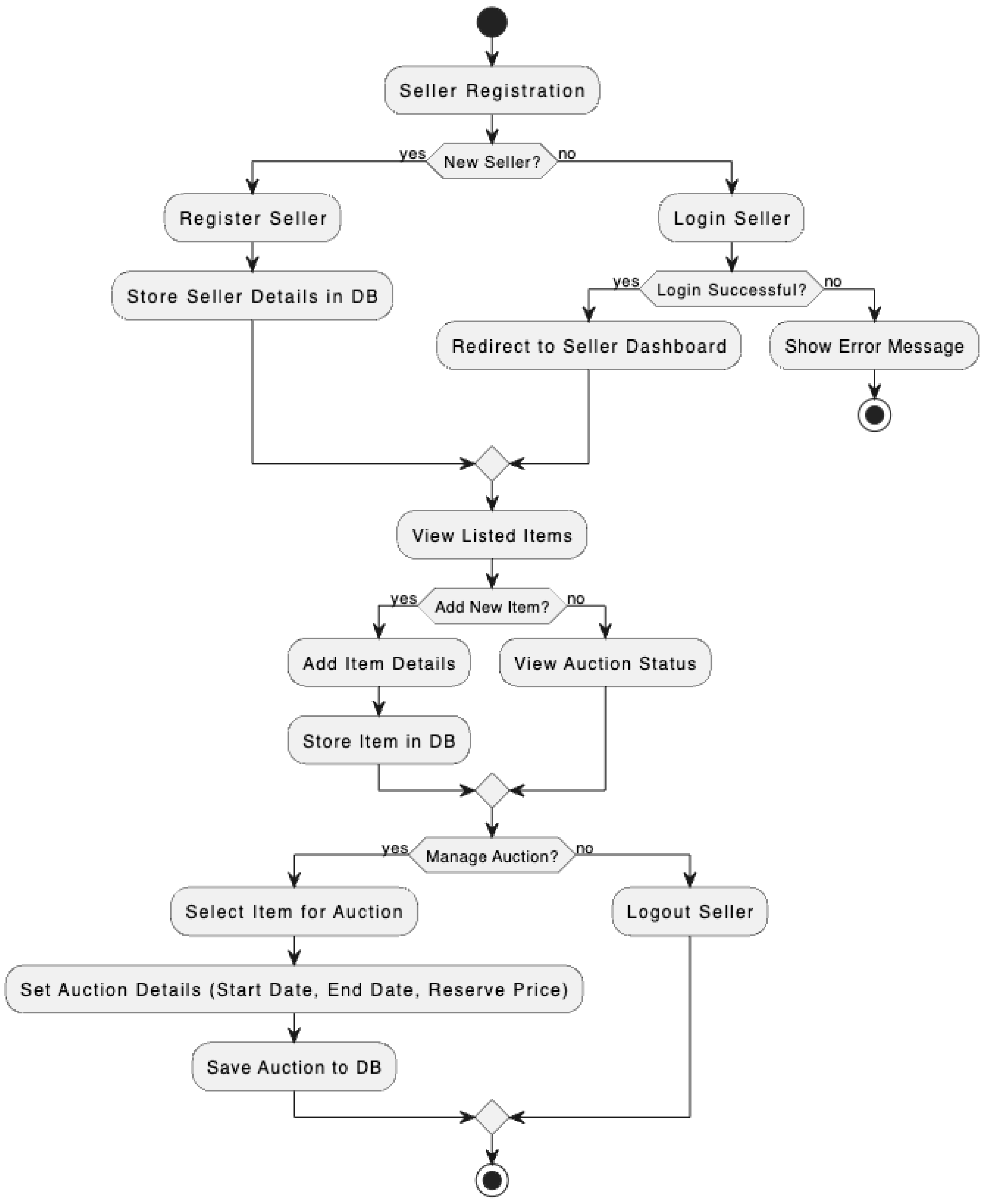
Analytics and Reporting: Vendors are provided with tools to monitor their sales performance, including total sales, the number of auctions that are successful, and feedback from buyers. This aids sellers in maximising the performance of their listings and pricing schemes going forward.

An online auction platform's seller management makes selling easier by offering these features, enabling sellers to manage their auctions profitably and efficiently.

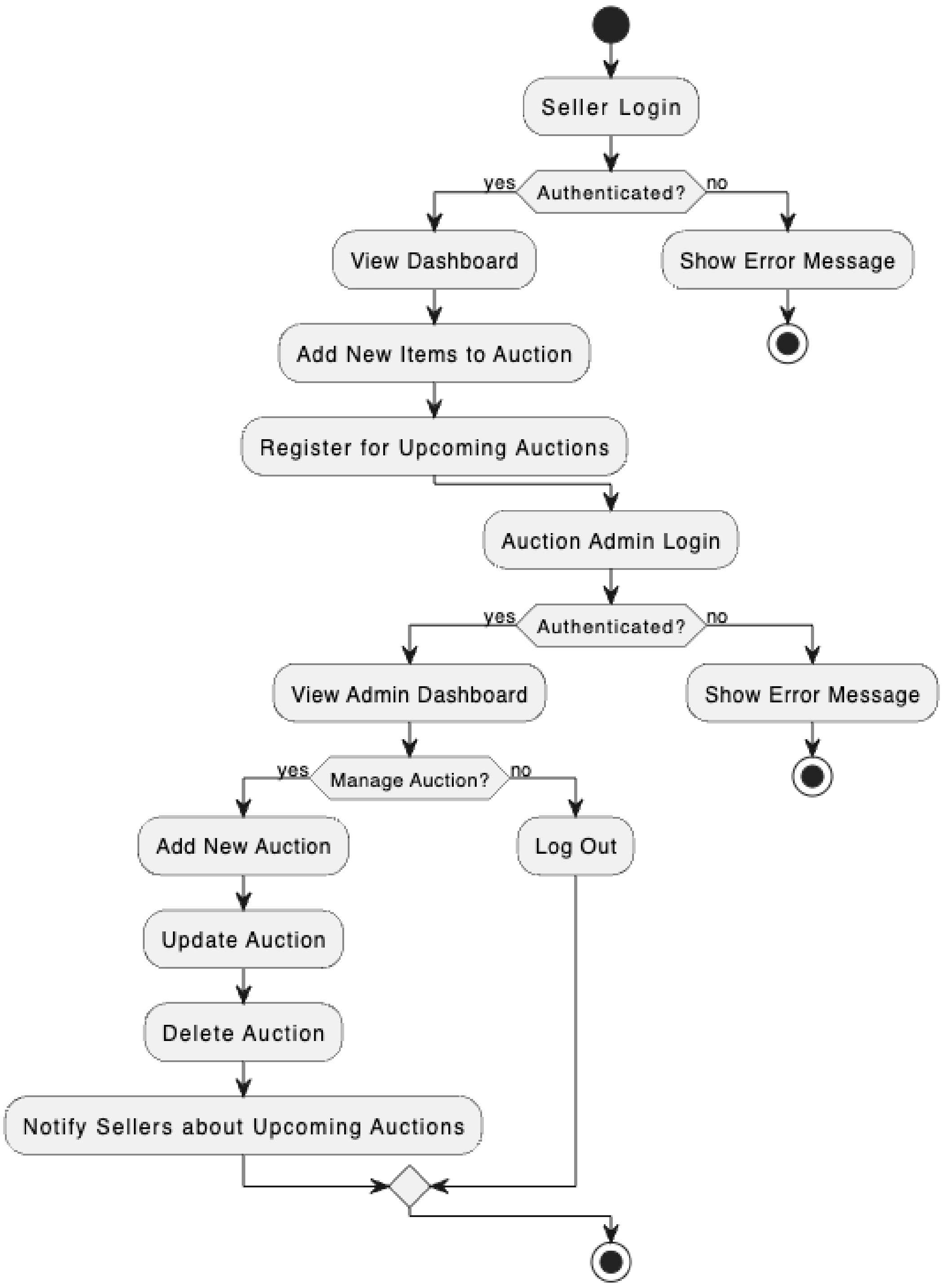
Completed Work	**Work in Progress**	**Work to be Completed**
Implementing the main user interface of the system	Implementing the security of the functions	Have to implement the forgot password function for users login
Implementing the seller account and signup		
Implementing the items listings and registration to the auctions form		
Implementing auctions, auction houses and categories pages.		
Implementing auction creation and updation and deletion processes.		
Implemented admin panel for auctions, items and sellers management.		
Implemented search and filtering options for items and auctions		

Flow chart

<u>Seller management</u>



Auction handling



Database Schemas:

Auction.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;


const AuctionSchema =  new Schema({

    title: {

        type: String,

        required: true

        },

    category:{

        type: String,

        required: true

    },

    description:{

        type: String,

        required: true

    },


    image:{

        type: String,

        required: false

    },

    items: [{


        type: Schema.Types.ObjectId,

        ref: 'Item' // Name of the model you are referring to


    }],
   startingDateTime: {
      type: Date, // Use the Date type for date and time
      required: true
   },
   location:{
      type: String,
      required: true

   },
   registeredBidder: [{
      type: Schema.Types.ObjectId,
       ref: "Bidder" }],

   registeredUsers: [{
      type: Schema.Types.ObjectId,
       ref: "Seller" }],


});


const Auction = mongoose.model("Auction",AuctionSchema);
module.exports = Auction;
```

Item.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;


const ItemSchema  = new Schema({


   name: {

      type: String,

      required: true

   },

   startingPrice: {

      type: Number,

      required: true

   },

   description: {

      type: String,

      required: true

   },


   category: {

      type: String,

      required: true

   },

   brand: {

      type: String,

      required: false

   },
features: {
    type: String,
    required: false
},
material: {
    type: String,
    required: false
},
auction: mongoose.Schema.Types.ObjectId,
condition:{
    type: String,
    required: false
},
images: [
    {
      data: Buffer, // Store image data as binary
      contentType: String // Store the image content type (e.g., 'image/jpeg')
    }
  ],
  registeredSeller: [{
    type: Schema.Types.ObjectId,
     ref: "Seller" }],


},
{
  timestamps: true,
// seller:{
//     type: Schema.Types.ObjectId,
//     ref: 'User',
//     required: true
// }
```

```
  });


  const Item = mongoose.model("Item",ItemSchema);

  module.exports = Item;
```

Seller.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;


const sellerSchema = new Schema({

  firstName: {

    type: String,

    required: true

  },

  lastName: {

    type: String,

    required: true

  },

  email: {

    type: String,

    required: true,

    unique:true

  },

  username: {

    type: String,

    required: true,

    unique:true

  },

  password: {
    type: String,
    required: true
},
country: {
    type: String,
    required: false
},
address: {
    type: String,
    required: true
},
companyName: {
    type: String,
    required: false
},
businessAddress: {
    type: String,
    required: false
},
contactInfo: {
    type: String,
    required: true,
    unique:true
},
paymentMethod: {
    type: String,
    required: true
},


birthday: {
    type: Date,
    required: true
    },




});


const Seller = mongoose.model("Seller",sellerSchema);

module.exports = Seller;
```

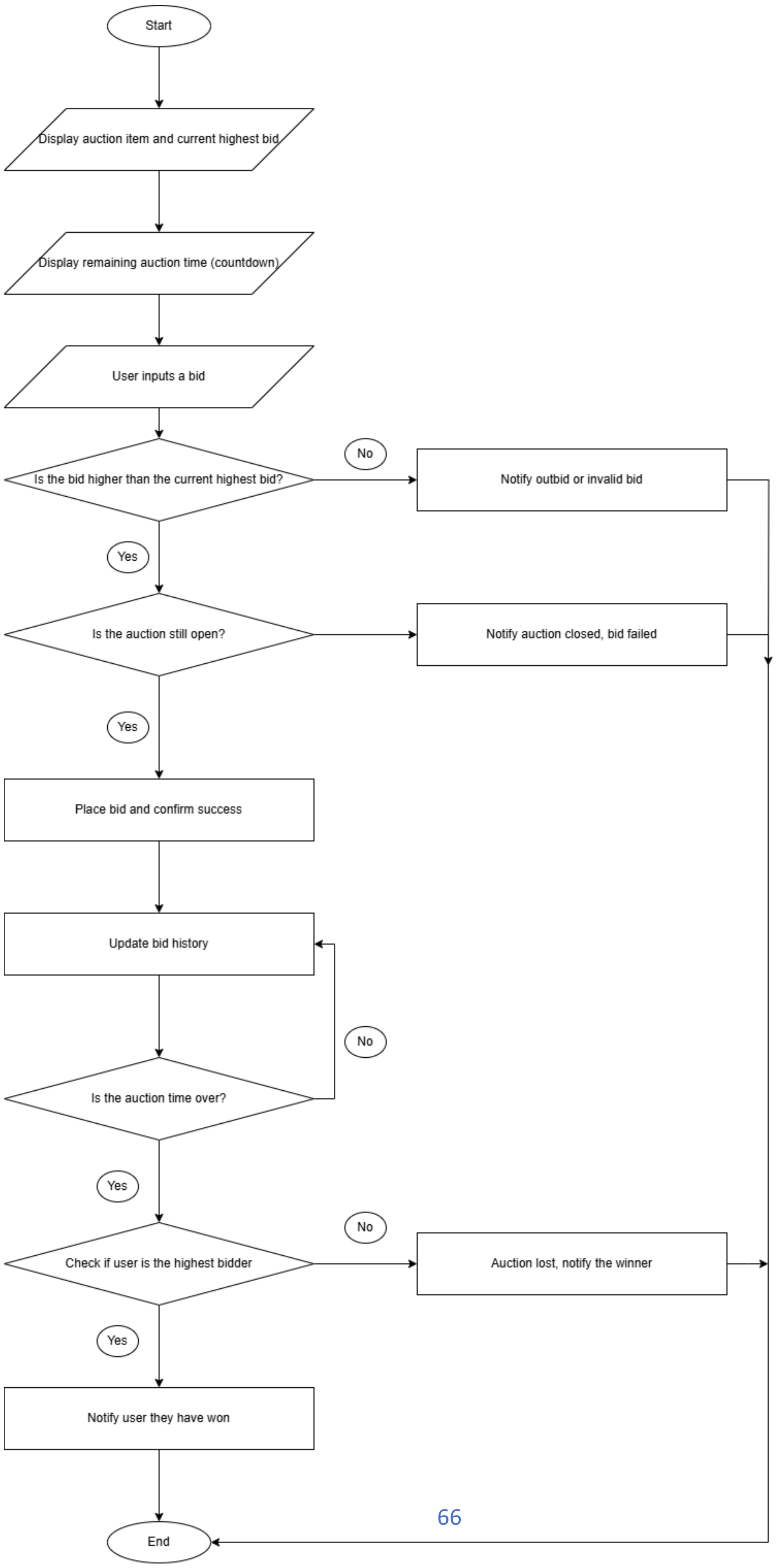
I. Bidding mechanism (IT22606174 – Samaranayake W.M.R.L)

A bidding mechanism system is designed to facilitate auctions by enabling participants to place bids on items. It ensures transparency, competitiveness, and fairness throughout the auction process. The system typically supports different auction types, such as open or sealed bids, and manages key functions like displaying items, tracking the highest bids in real-time, setting bid increments, and monitoring auction timers. Users can place bids, receive notifications on bid status, and get alerts at auction milestones, such as being outbid or winning. The system ensures that bids are placed securely and in compliance with auction rules.

Work Progression

Completed Work	Work in Progress	Work to be Completed
• Implement CRUD operations with validations for Bids, Auction Items, and Bidder Profiles database collections. • Implementing a function to show the current highest bid on an auction item. • Implementing confirmation notifications for successfully placed bids. • Linking bid placement with auction end times and restricting bidding when a user attempts to outbid themselves	• Auction Countdown Timer to display the remaining time for the auction. • Outbid Notification System to notify bidders when they have been outbid during an auction.	• Auction Win Notification to alert the highest bidder at the auction's end. • Auto-Bid Feature Implementation to automatically increase bids on behalf of the bidder up to a set limit.

Flow chart


Start
Display auction item and current highest bid
Display remaining auction time (countdown)
User inputs a bid
Is the bid higher than the current highest bid?
No
Notify outbid or invalid bid
Yes
Is the auction still open?
Notify auction closed, bid failed
Yes
Place bid and confirm success
Update bid history
No
Is the auction time over?
Yes
No
Check if user is the highest bidder
Auction lost, notify the winner
Yes
Notify user they have won
End

Algorithm:

Start

Step 1: Display Auction Item

- Show the details of the auction

- item and the current highest bid.

Step 2: Input Bid

-Prompt the user to input their bid amount.

Step 3: Check Bid Validity

- If the bid is greater than the current highest bid, go to Step 4.

- Else, notify the user that the bid is outbid or invalid and return to Step 2.

Step 4: Place Bid

- Update the highest bid with the new bid amount.
- Confirm the success of the bid placement to the user.

Step 5: Check Auction Status

- If the auction is still open, continue to Step 6.
- Else, notify the user that the auction has closed, and the bid has failed, then go to Step 12.

Step 6: Update Bid History

-Store the details of the bid in the bid history.

Step 7: Display Remaining Time

-Show a countdown of the remaining time for the auction.

Step 8: Check Auction Time

-If the auction time is over, go to Step 9.

Else, return to Step 2 for more bids.

Step 9: Check Winner

-Determine if the user is the highest bidder at the auction's end.

Step 10: Notify User

- If the user is the highest bidder, notify them that they have won.

  - Else, notify the user that they lost the auction and inform the winning bidder.

End

Pseudocode:

```
START

// Step 1: Display Auction Item
DISPLAY auction_item_details
DISPLAY current_highest_bid

WHILE auction_is_open DO

  // Step 2: Input Bid
  PROMPT user "Enter your bid amount:"
  INPUT user_bid

  // Step 3: Check Bid Validity
  IF user_bid <= current_highest_bid THEN
    NOTIFY user "Your bid is outbid or invalid. Please enter a higher bid."
    CONTINUE // Go back to Step 2

  // Step 4: Place Bid
  current_highest_bid = user_bid
  NOTIFY user "Your bid has been placed successfully."

  // Step 5: Check Auction Status
  IF auction_is_still_open THEN

    // Step 6: Update Bid History
    ADD user_bid to bid_history

    // Step 7: Display Remaining Time
```

```
        DISPLAY countdown_timer


        // Step 8: Check Auction Time
        IF countdown_timer is over THEN
            BREAK // Exit the loop to go to Step 9
        ENDIF
    ELSE
        NOTIFY user "The auction has closed."
        BREAK // Exit the loop to go to Step 12
    ENDIF

ENDWHILE

// Step 9: Check Winner
IF user is highest_bidder THEN
    // Step 10: Notify User
    NOTIFY user "Congratulations! You have won the auction."
ELSE
    NOTIFY user "You have lost the auction."
ENDIF

// Step 11: End
END
```

Database Schemas:

Item Model

```
{
  "_id": ObjectId,
  "name": String,
  "description": String,
  "startingPrice":
Number,
  "currentHighestBid":
Number,

"currentHighestBidder":
ObjectId,
  "auctionEndTime":
Date,
  "createdDate": Date
}
```

Bid Model

```
{

  "_id": ObjectId,

  "itemId": ObjectId

  "userId": ObjectId,

  "amount": Number,

  "timestamp": Date
```

Advertising Handling System(IT22224170-Sanjana K.G.T.S)

The Advertising Handling System serves as the central hub for managing advertisements throughout their lifecycle within the organization. Under the supervision of the advertisement manager, this system ensures a streamlined process for creating, updating, and deleting advertisements, thus maintaining the integrity of the advertising database.

Advertisement Creation involves user-friendly interfaces that allow users to input detailed information about each advertisement, ensuring comprehensive and accurate data capture. The Advertisement Update functionality enables quick modifications to be existing ads, allowing for timely adjustments in response to market trends or advertiser requests. Advertisement Deletion provides a mechanism to remove outdated or incorrect advertisements, keeping the database relevant and reliable.

The system also features an Advertisement Search capability, facilitating easy retrieval of ads based on various criteria, which enhances user efficiency and ensures that relevant advertisements are easily accessible.

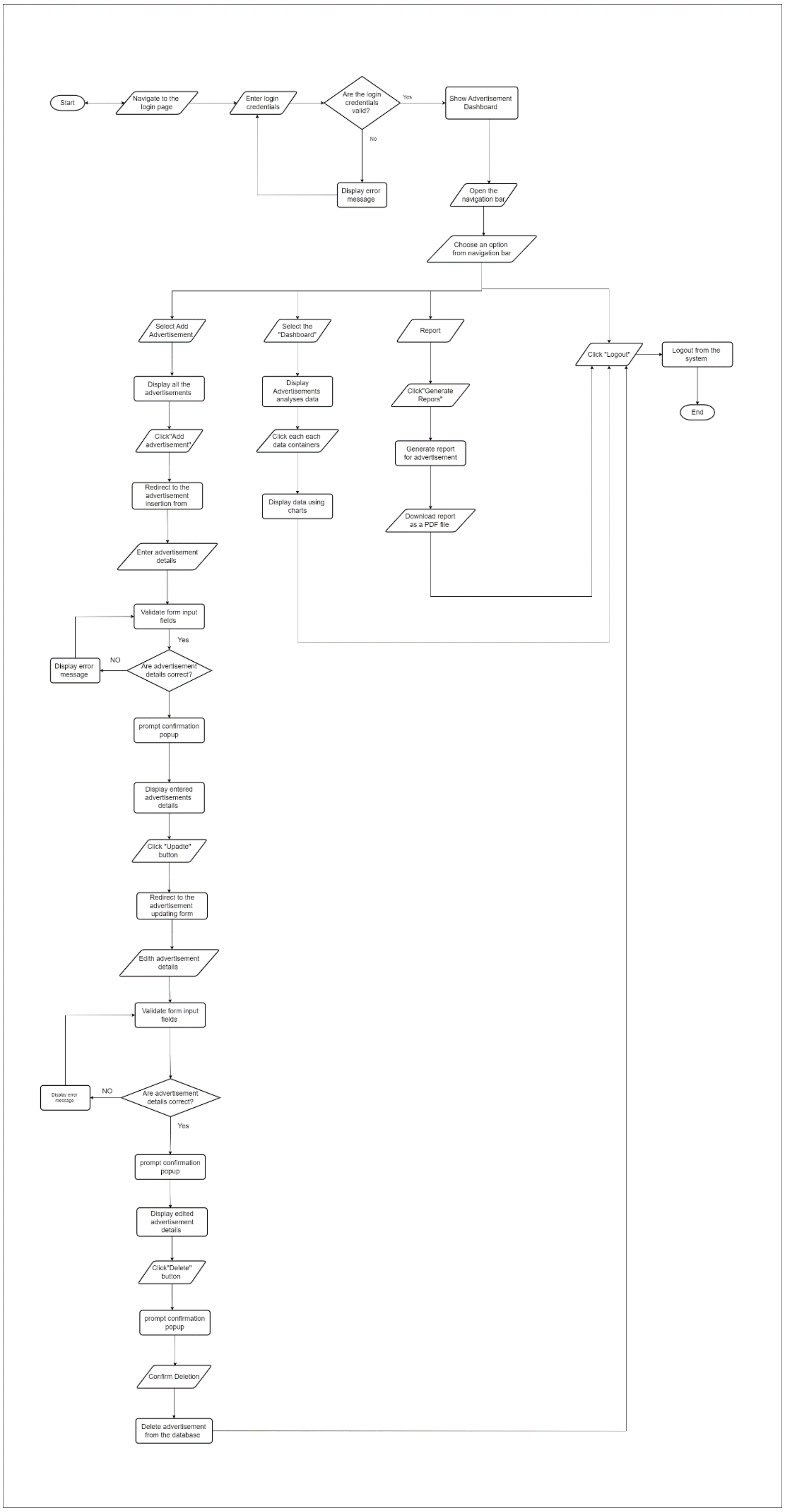
To support strategic decision-making, the system incorporates a Report Generation feature that compiles advertising data into detailed reports. These reports provide essential insights into advertising performance and effectiveness.

Moreover, the Analytics Generation function visualizes key advertising metrics through interactive bar graphs and pie charts, allowing stakeholders to quickly grasp trends and patterns in advertisement performance. This comprehensive approach ensures that the Advertising Handling System effectively supports advertising management while providing valuable insights for optimization and strategic planning.

Work Progression

Completed Work	Work in Progress	Work to be Completed
• Implementing CRUD operations with validations for Advertisements database collections. • Implementing a search function to search Advertisements by the Title and advertisement date. • Implementing a function to generate reports for Advertisements Records. • Complete half of Advertisement Data Analytics Generation.	• Incomplete other half of the data analytics Generation (Pie chart, Bar chart)	• Published the created Advertisements to the social media (Facebook)

Flow chart



Start
Navigate to the login page
Enter login credentials
Are the login credentials valid?
Yes
No
Show Advertisement Dashboard
Display error message
Open the navigation bar
Choose an option from navigation bar
Select Add Advertisement
Select the "Dashboard"
Report
Click "Logout"
Logout from the system
End
Display all the advertisements
Display Advertisements analyses data
Click"Generate Repors"
Click"Add advertisement"
Click each each data containers
Generate report for advertisement
Redirect to the advertisement insertion from
Display data using charts
Download report as a PDF file
Enter advertisement details
Validate form input fields
Yes
NO
Display error message
Are advertisement details correct?
prompt confirmation popup
Display entered advertisements details
Click "Upadte" button
Redirect to the advertisement updating form
Edith advertisement details
Validate form input fields
Display error message
NO
Are advertisement details correct?
Yes
prompt confirmation popup
Display edited advertisement details
Click"Delete" button
prompt confirmation popup
Confirm Deletion
Delete advertisement from the database

Algorithm:

Step 1: Start

Step 2: Access BIDMASTER web application

Step 3: Login to the system

a. If the entered credentials are incorrect
   - Display "Entered username and password don't match"
   - Return to Step 3

b. If the entered credentials are correct
   - Proceed to Step 4

Step 4: Display the Dashboard after logging in

Step 5: Open the Navigation Bar

Step 6: Choose an option from the navigation bar

a. If "Advertisement Details" was selected
   - Display all the advertisements in the advertisement database collection

a2. If "Add Advertisement" button is clicked
   - Redirect to the advertisement creation form

a3. If an Advertisement name is clicked

- Display all the data for the relevant advertisement

a3.1. If “Edit” button is clicked

- Redirect to the advertisement updating page

a3.2. If “Delete” button is clicked

- Delete advertisement from the database

b. If “Dashboard” was selected
   - Display all the advertisements record’s using pie chart, Bar charts

c. If “Reports” was selected
   - Generate advertisement record report
   - Download the report as a PDF

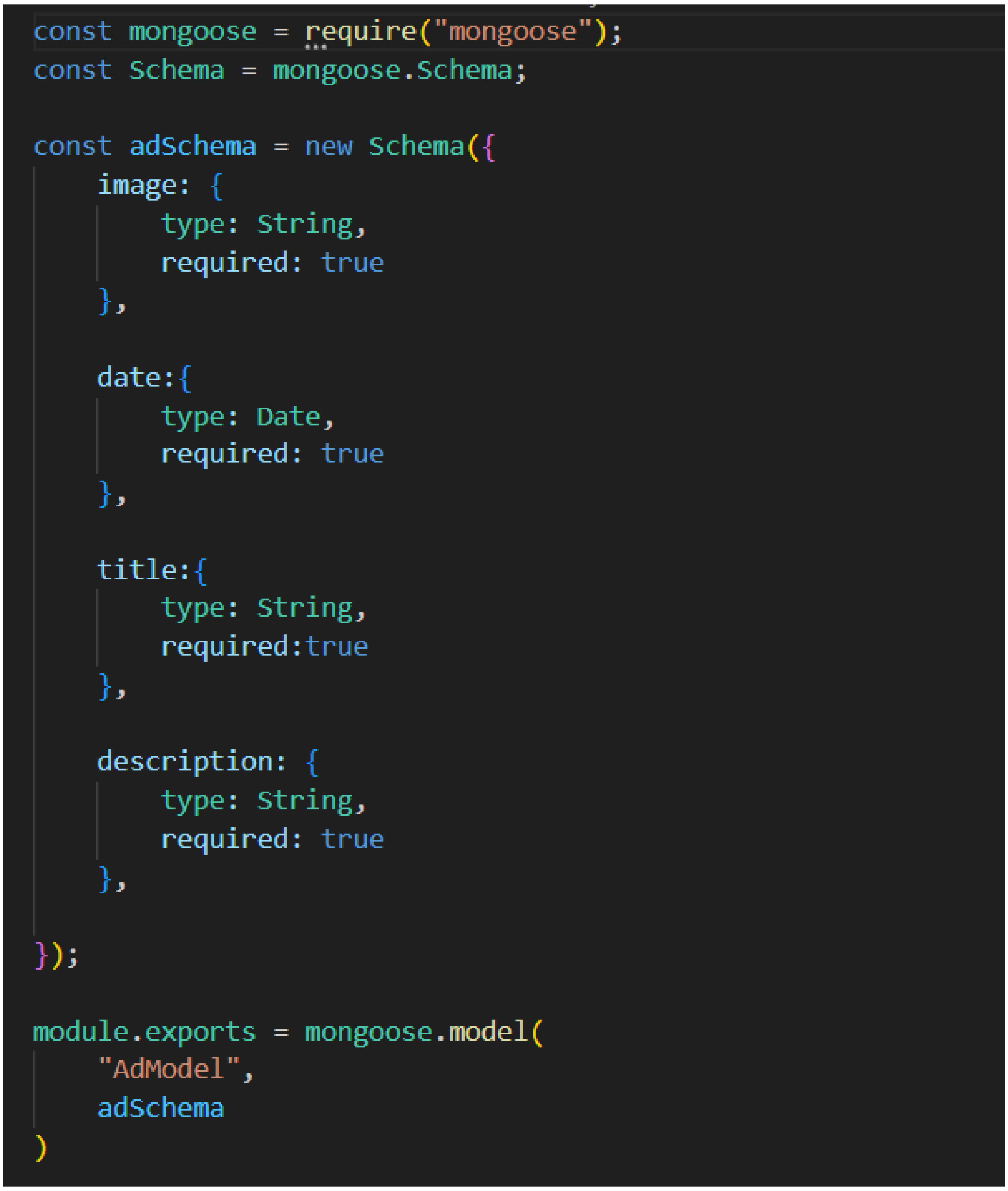
Step 7: Logout of the system

End

Pseudocode:

```
Start

Access BIDMASTER web application

Login to the system
    If entered credentials are incorrect
        Display “Entered username and password don’t match”
        Go back to Login
    Else
    Proceed to Display the Dashboard

Display the Dashboard

Select from the Navigation Bar

Choose an option from the navigation bar
  If "Advertisement details" was selected
      Display all advertisements in the database collection

  If "Add Advertisement" button is clicked
      Redirect to advertisement creation form

  If search for an advertisement Title/Date
      Filter and display relevant results

  If an advertisement name is clicked
      Display all data for the relevant advertisement

  If "Edit" button is clicked
      Redirect to the advertisement updating page

  If "Delete" button is clicked
      Delete advertisement from the database

  If "Dashboard" was selected
      Display all the advertisements record's using pie chart, Bar charts
    If "Reports" was selected
        Generate advertisement record report
        Download the report as a PDF

Logout of the system

End
```

Database Schemas



```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const adSchema = new Schema({
    image: {
        type: String,
        required: true
    },

    date:{
        type: Date,
        required: true
    },

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

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

});

module.exports = mongoose.model(
    "AdModel",
    adSchema
)
```

Registration management (IT22215192 - Viduranga S P S)

In the User Registration Management role, the registration function is a critical component of the overall user experience, managing the flow from initial sign-up to full system access. It ensures the efficient coordination of several key elements, such as user sign-up, validation, profile creation, and role-specific navigation, all under the supervision of the registration manager.

This function involves the meticulous maintenance of a comprehensive user database, ensuring accurate real-time logging and updating of user details, including name, email, contact information, and role. It also keeps track of the registration status and profile updates.

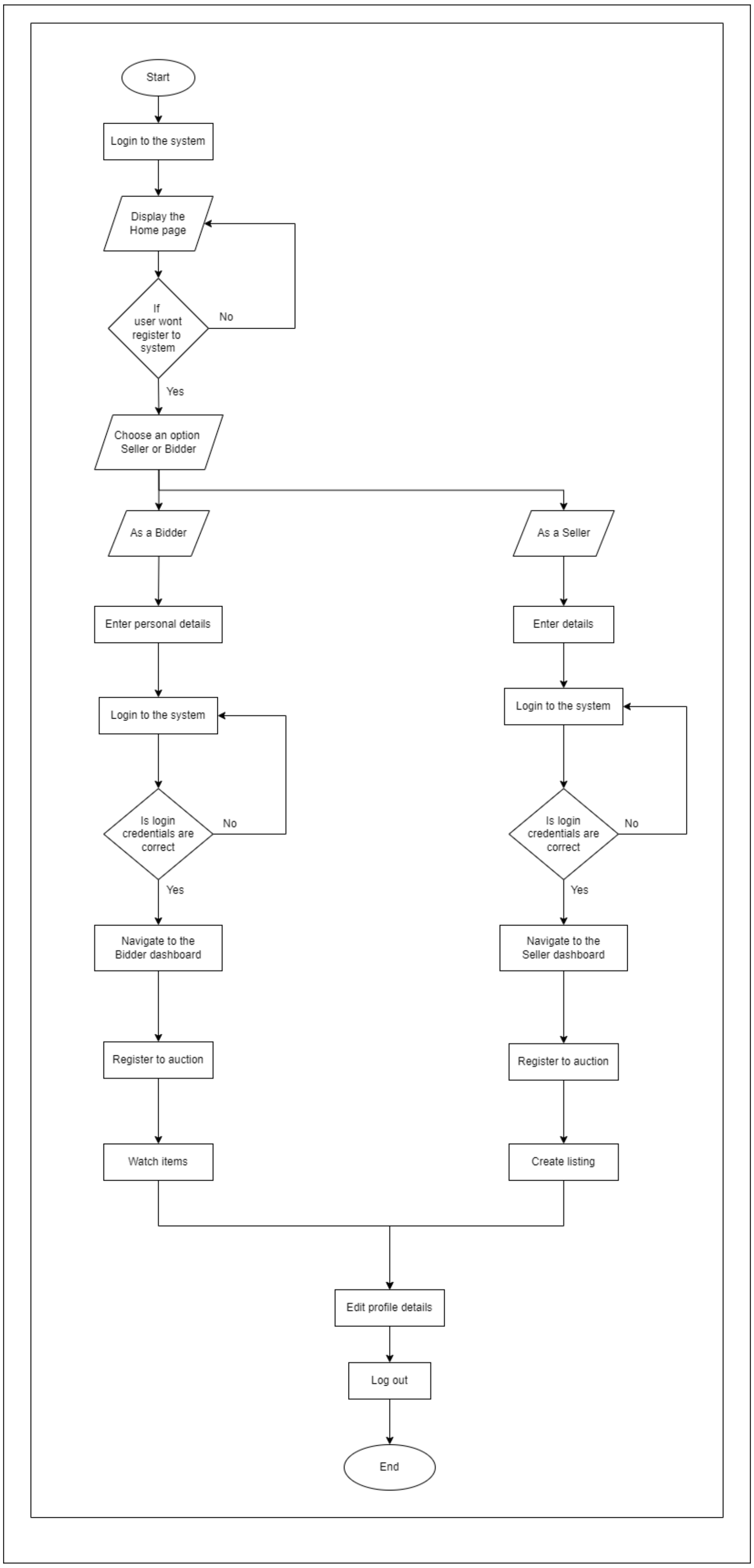
From initial registration to final profile confirmation, the system closely monitors each user's status. To provide real-time updates on registration progress, role-specific features, and upcoming auctions or listing options, it integrates email notifications and system alerts. Each user interaction is tracked through a dynamic system that automatically updates their profile with milestones like auction registration or listing creation.

Based on user activity, the system generates detailed reports that provide key insights into registration trends, role-specific actions, and any issues or delays encountered during the registration process. These reports are essential tools for improving registration efficiency, monitoring user engagement, and enabling data-driven decision-making for the platform.

Work Progression

Completed Work	Work in Progress	Work to be Completed
Implementing CRUD operations with validations for users in the registration database. Implementing a search function to search users by name or email. Implementing a function to generate reports for user registrations. Linking users, registered auctions, and profiles with other functions, such as auction participation for bidders.	Improving system registration security to ensure each user's details are only visible to the relevant user.	Display a real-time dashboard for the registration manager to monitor registration trends and active users, including a pie chart of user roles.

Flow chart


Start
Login to the system
Display the Home page
If user wont register to system
No
Yes
Choose an option Seller or Bidder
As a Bidder
As a Seller
Enter personal details
Enter details
Login to the system
Login to the system
Is login credentials are correct
No
Yes
Is login credentials are correct
No
Yes
Navigate to the Bidder dashboard
Navigate to the Seller dashboard
Register to auction
Register to auction
Watch items
Create listing
Edit profile details
Log out
End

Algorithm:

Step 1: Start
Step 2: Access the Bidmaster web application
Step 3: Click the "Sign Up" button
Step 4: Fill in the registration form with user information (name, Email, Contact information, Gender, Date of birth, Address, Username, Password).

a. If any required fields are not filled or if incorrect data is entered

- Show error messages in the corresponding fields.
- Return to Step 4
  b. If all fields are correctly filled
- Proceed to Step 5
  Step 5: Submit the registration form
  a. If the registration is successful
- Display "Registration successful, please log in"
- Return to Step 2
  b. If there is a problem with registration
- Display the error message
- Return to Step 4

  Step 6: After registration, users can log in with their credentials
  Step 7: Show the user dashboard once logged in
  Step 8: Users can view and update their profile details or registered auctions.
  Step 9: Logout of the system (if needed)
  End

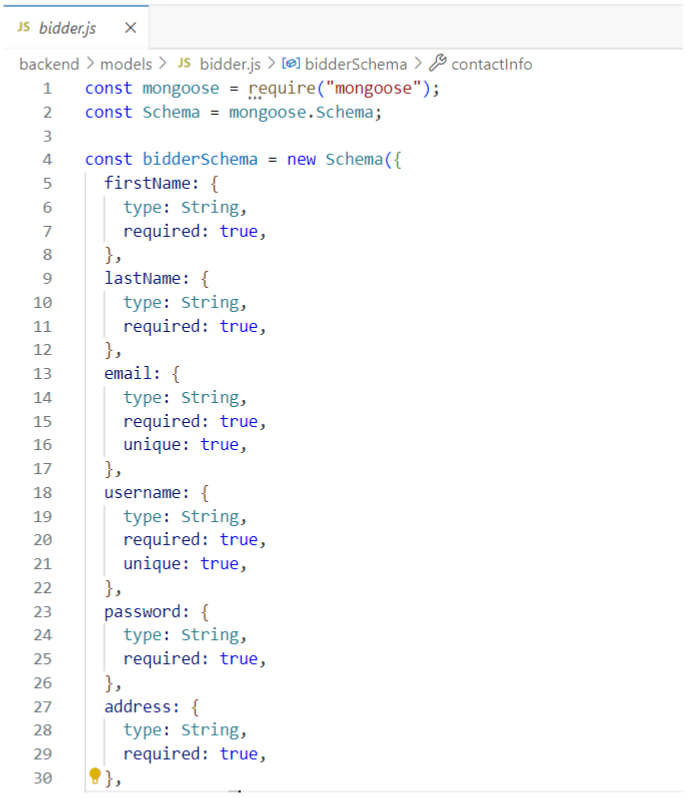
Pseudocode:

```
BEGIN
Step 1: Start the application
Step 2: Access Bidmaster web application
Step 3: Click "Sign Up" button
  DISPLAY registration form
Step 4: Fill in registration form (name, Email , Contact information, Gender, Date of birth, Address,
Username, Password)
  IF any required fields are empty or incorrect THEN
    DISPLAY error messages for corresponding fields
```

```
        RETURN to Step 4
    ELSE
        GO TO Step 5
    END IF
Step 5: Submit the registration form
    IF registration is successful THEN
        DISPLAY "Registration successful, please log in"
        RETURN to Step 2
    ELSE
        DISPLAY registration error message
        RETURN to Step 4
    END IF
Step 6: After registration, log in with credentials
Step 7: Show user dashboard once logged in
    IF user is a bidder THEN
        DISPLAY auction items and auction registration options
    ELSE IF user is a seller THEN
        DISPLAY listing options
    END IF
Step 8: Users can view and update their profile details or registered auctions
Step 9: Logout if necessary
END
```

Database Schemas:

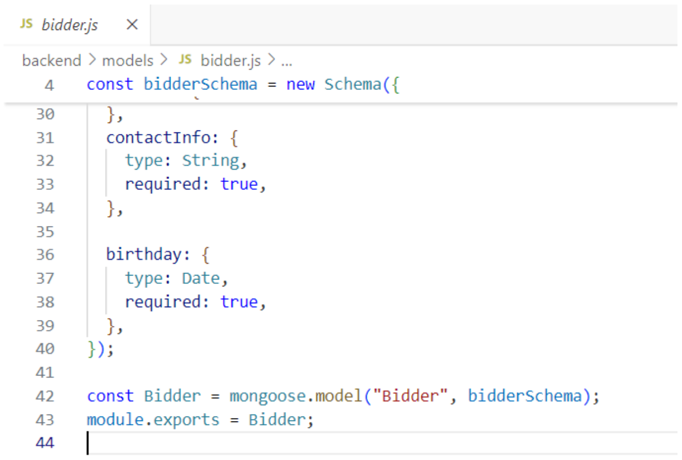
Bidder model



bidder.js ×

backend > models > bidder.js > bidderSchema > contactInfo

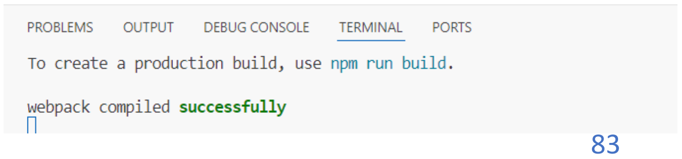
```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const bidderSchema = new Schema({
  firstName: {
    type: String,
    required: true,
  },
  lastName: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  username: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  address: {
    type: String,
    required: true,
  },
```



bidder.js ×

backend > models > bidder.js > ...

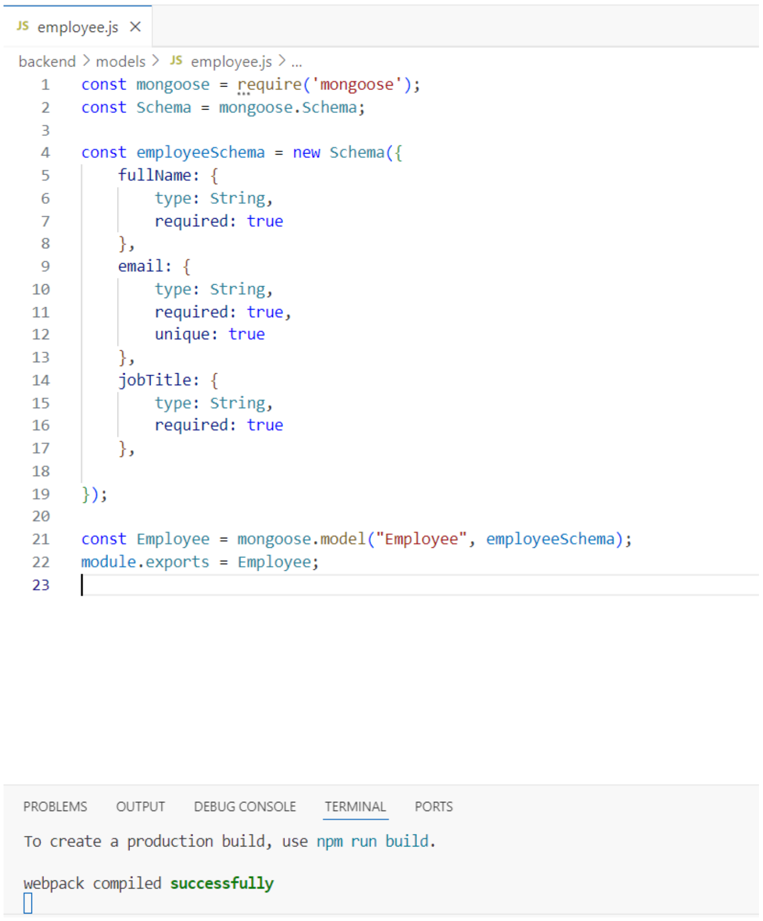
```
const bidderSchema = new Schema({
  },
  contactInfo: {
    type: String,
    required: true,
  },

  birthday: {
    type: Date,
    required: true,
  },
});

const Bidder = mongoose.model("Bidder", bidderSchema);
module.exports = Bidder;

```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
To create a production build, use npm run build.

webpack compiled successfully
```

Employee model



```javascript
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const employeeSchema = new Schema({
    fullName: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    jobTitle: {
        type: String,
        required: true
    },

});

const Employee = mongoose.model("Employee", employeeSchema);
module.exports = Employee;

```

```
To create a production build, use npm run build.

webpack compiled successfully
```

Payment Gateway (IT22091970 Chandraguptha H.A.T)

Payment Flow Coordination – As the payment gateway service provider, you will handle the flow of payments from buyers to sellers and shipping providers. Payment for shipping services will be linked directly to the item purchase.

Security Measures – Ensure that payment data (e.g., credit card information) is encrypted and that transactions are secure. Use secure APIs (such as Stripe, PayHere) for payment handling.

Integration with Delivery System – The payment gateway should be integrated with the delivery management system to track when items are marked as "Delivered" so payments can be released to the seller and shipping provider.

Reporting – Generate reports for payments made, payments pending, and any transaction errors to maintain transparency.

Completed Work	Work in Progress	Work to be Completed
Implemented payment gateway integration with item purchases for both buyers and shipping providers.	Enhancing payment security by implementing encryption for credit card data and transactions.	Implement automatic payment release feature based on "Delivered" status in the delivery system.
Successfully integrated the payment system with the delivery management system to track delivery status.	Testing and validating payment flow for various scenarios (successful, failed, pending payments).	Generate detailed payment reports for both sellers and shipping providers for transparency.
Developed basic transaction logging for completed payments.	Improving API error handling for failed transactions with proper notifications to the buyer and seller.	Implement real-time payment status monitoring and update for both sellers and shipping providers.

Algorithm:

1. **Start**
2. **Access Bidmaster web application**
3. **Login to the system**
   - **If credentials are incorrect:** Display error and retry login.
   - **If correct credentials:**
     - **If seller:** Proceed to Step 4.
     - **If shipping provider:** Proceed to Step 8.
4. **Seller's Dashboard**:
   - **Select "Generate PDF"**: Generate report as a PDF.
   - **Select "Add Delivery Person"**: Complete and submit the form.
     - If errors, display messages and retry.
     - If successful, display "Delivery person added."
5. **Shipping Provider's Dashboard**:
   - **Select "Enter Delivery Details"**: Complete and submit the form with delivery date, time, and status.
     - If errors, display messages and retry.
     - If successful, display "Delivery details entered."
6. **Logout (optional)**
7. **End**

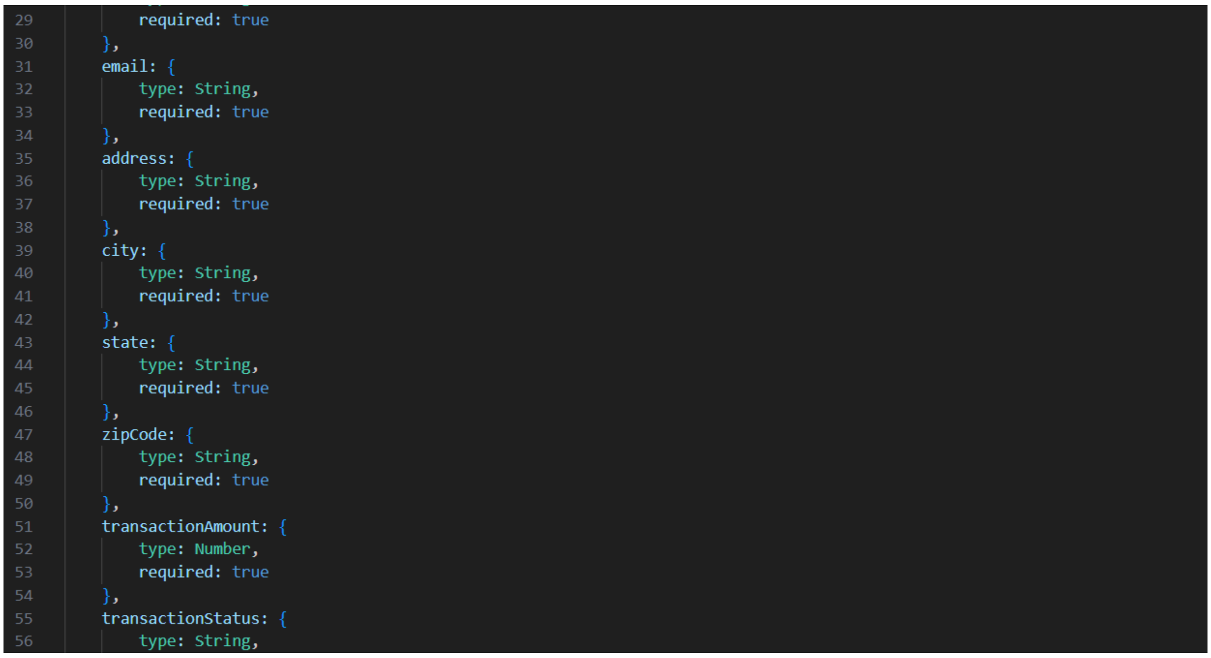
pseudocode

```
BEGIN
Step 1: Start the application
Step 2: Access Bidmaster web application
Step 3: Login to the system
  IF credentials are incorrect THEN
    DISPLAY "Entered username and password don’t match"
    RETURN to Step 3
  ELSE
```

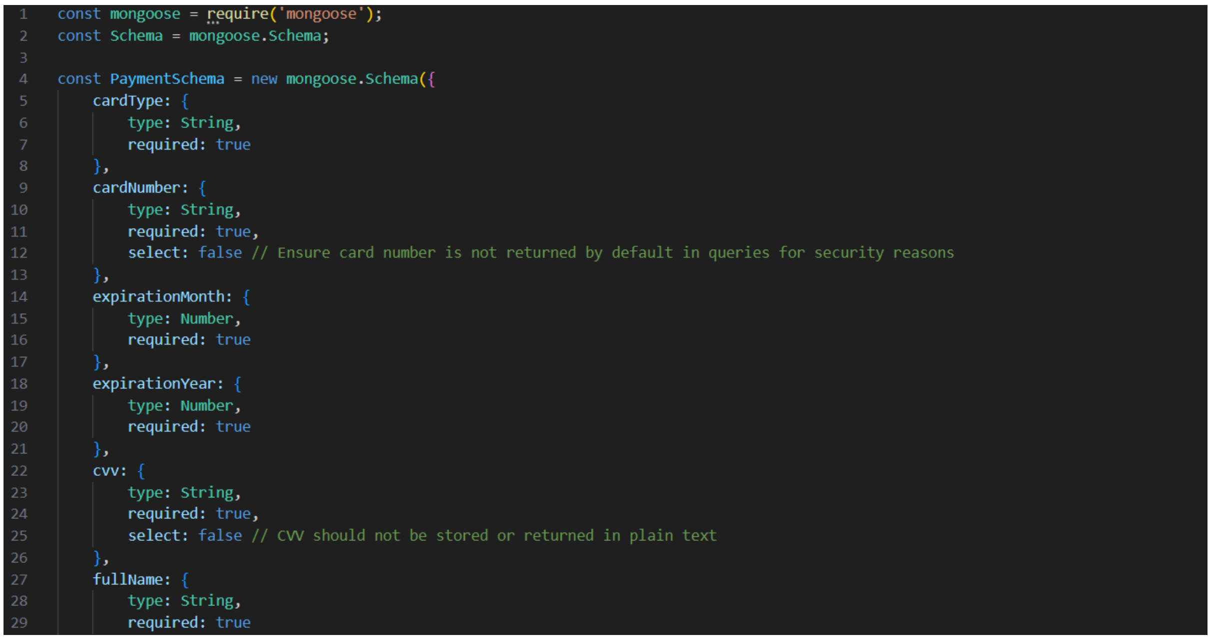
```
        IF user is a seller THEN
            GO TO Step 4
        ELSE IF user is a shipping provider THEN
            GO TO Step 7
        END IF
    END IF

Step 4: Display seller's Dashboard
    IF payment processing for shipping service is selected THEN
        PROCESS payment through the payment gateway
        IF payment is successful THEN
            CONFIRM payment to the seller and shipping provider
        ELSE
            DISPLAY payment error
        END IF
    END IF

Step 5: Manage transaction status
    IF "Track Payment Status" is selected THEN
        DISPLAY payment and delivery tracking details
    END IF

Step 6: IF payment confirmation for item delivery is received THEN
    RELEASE payment to seller and shipping provider
    DISPLAY "Payment released successfully"
    GO TO Step 12

Step 7: Display shipping provider's Dashboard
    IF "Process Delivery Payment" is selected THEN
        VERIFY shipping details and initiate payment
        IF successful THEN
```

```
            CONFIRM payment with the seller and buyer
        ELSE
            DISPLAY error message
        END IF
    END IF

Step 12: Logout if necessary

END
```

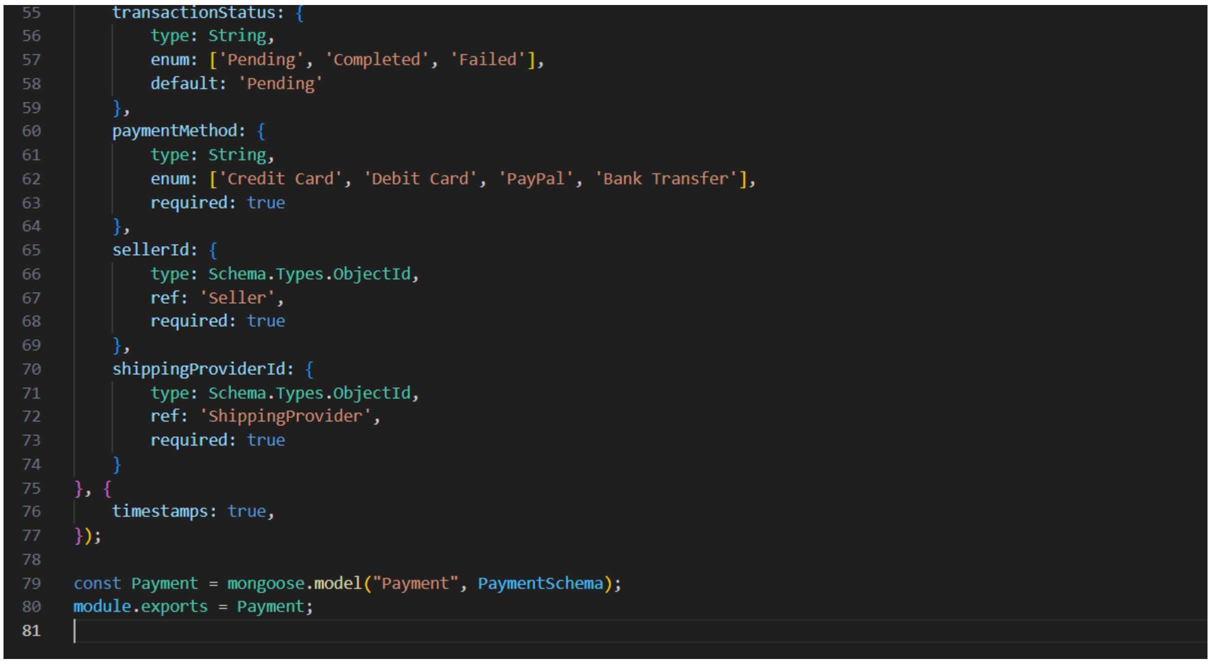
Database Schemas:



```
            required: true
        },
        email: {
            type: String,
            required: true
        },
        address: {
            type: String,
            required: true
        },
        city: {
            type: String,
            required: true
        },
        state: {
            type: String,
            required: true
        },
        zipCode: {
            type: String,
            required: true
        },
        transactionAmount: {
            type: Number,
            required: true
        },
        transactionStatus: {
            type: String,
```



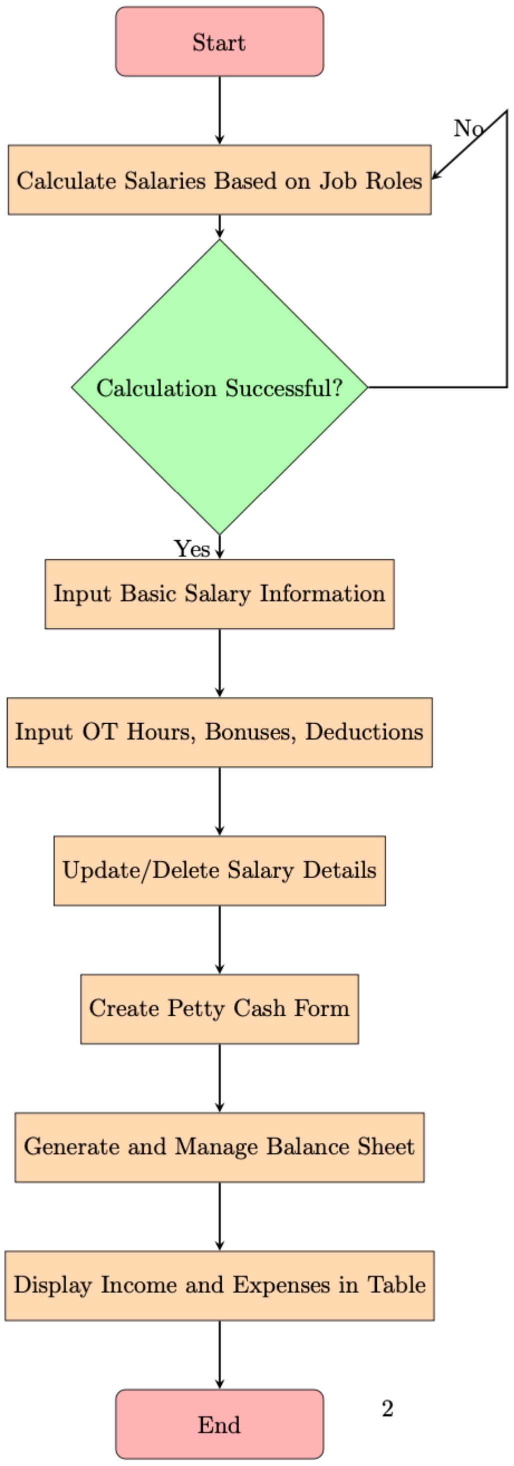
```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const PaymentSchema = new mongoose.Schema({
    cardType: {
        type: String,
        required: true
    },
    cardNumber: {
        type: String,
        required: true,
        select: false // Ensure card number is not returned by default in queries for security reasons
    },
    expirationMonth: {
        type: Number,
        required: true
    },
    expirationYear: {
        type: Number,
        required: true
    },
    cvv: {
        type: String,
        required: true,
        select: false // CVV should not be stored or returned in plain text
    },
    fullName: {
        type: String,
        required: true
```


```
    transactionStatus: {
        type: String,
        enum: ['Pending', 'Completed', 'Failed'],
        default: 'Pending'
    },
    paymentMethod: {
        type: String,
        enum: ['Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer'],
        required: true
    },
    sellerId: {
        type: Schema.Types.ObjectId,
        ref: 'Seller',
        required: true
    },
    shippingProviderId: {
        type: Schema.Types.ObjectId,
        ref: 'ShippingProvider',
        required: true
    }
}, {
    timestamps: true,
});

const Payment = mongoose.model("Payment", PaymentSchema);
module.exports = Payment;
```

Finance Handling system- **Pihillegedara S.N.M IT22238580**

Completed Work	Work in Progress	Work to be Completed
Calculated salaries for all employees based on their job roles.	Enhancing the salary calculation system to include bonuses and deductions.	Automate monthly salary generation and payment processing.
Created petty cash forms for routine expenses (e.g., office supplies, maintenance).	Validating petty cash reports with actual expenses to ensure accuracy.	Implement automated petty cash reconciliation and approval workflow.
Generated initial balance sheets by compiling income and expenses.	Testing the accuracy of balance sheet calculations, ensuring correct categorization of income and expenses.	Develop advanced financial reporting that includes cash flow statements and profit & loss accounts.

Flow chart



Start
Calculate Salaries Based on Job Roles
No
Calculation Successful?
Yes
Input Basic Salary Information
Input OT Hours, Bonuses, Deductions
Update/Delete Salary Details
Create Petty Cash Form
Generate and Manage Balance Sheet
Display Income and Expenses in Table
End

2

Algorithm:

Start

Access Bidmaster web application

Login to the system

If credentials are incorrect: Display error and retry login.

If correct credentials:

If financial manager: Proceed to Step 4.

Financial Manager's Dashboard:

Select "Calculate Salaries":

Input the employee job roles and calculate salaries.

If errors, display messages and retry.

If successful, display "Salaries calculated."

Select "Create Petty Cash Form":

Input petty cash details such as amount, date, and purpose.

If errors, display messages and retry.

If successful, display "Petty cash form created."

Select "Create Balance Sheet":

Enter income and expenses for the period.

Generate the balance sheet.

If errors, display messages and retry.

If successful, display "Balance sheet created."

Logout (optional)

End

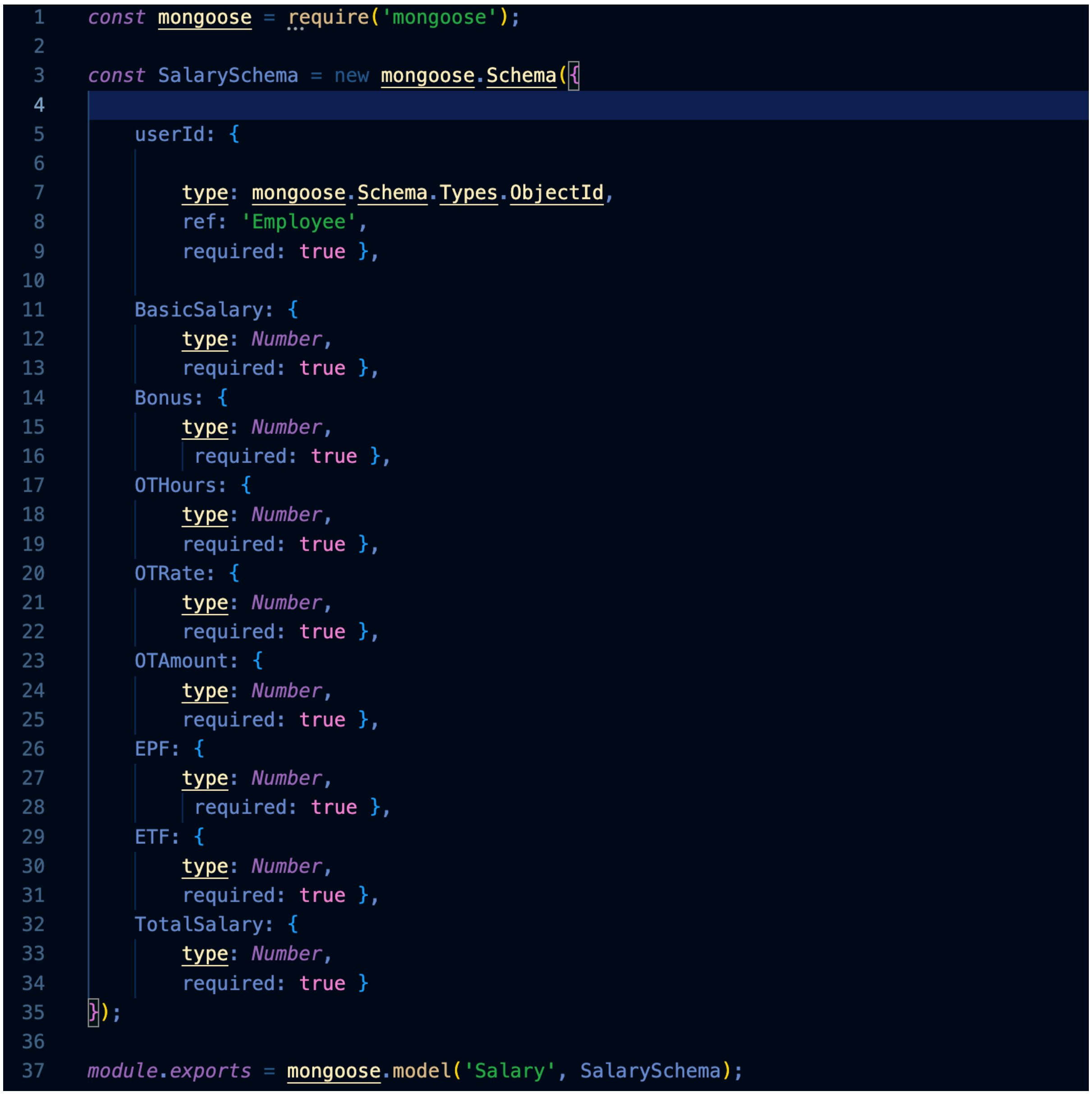
Pseudocode

```
BEGIN


Step 1: Start the application


Step 2: Access Bidmaster web application
Step 3: Login to the system
  IF credentials are incorrect THEN
    DISPLAY "Entered username and password don’t match"
    RETURN to Step 3
  ELSE IF user is a financial manager THEN
    GO TO Step 4
  ELSE
    DISPLAY "Access Denied"
    RETURN

Step 4: Display Financial Manager's Dashboard
  WHILE application is running DO
    DISPLAY options: [Calculate Salaries, Create Petty Cash Form, Create Balance Sheet, Logout]

    GET user selection

    IF selection is "Calculate Salaries" THEN
      GO TO Step 5
    ELSE IF selection is "Create Petty Cash Form" THEN
      GO TO Step 8
    ELSE IF selection is "Create Balance Sheet" THEN
      GO TO Step 11
    ELSE IF selection is "Logout" THEN
      GO TO Step 12
    END IF
  END WHILE

Step 5: Calculate Salaries
  FOR each employee in the system DO
    GET employee's job role
```

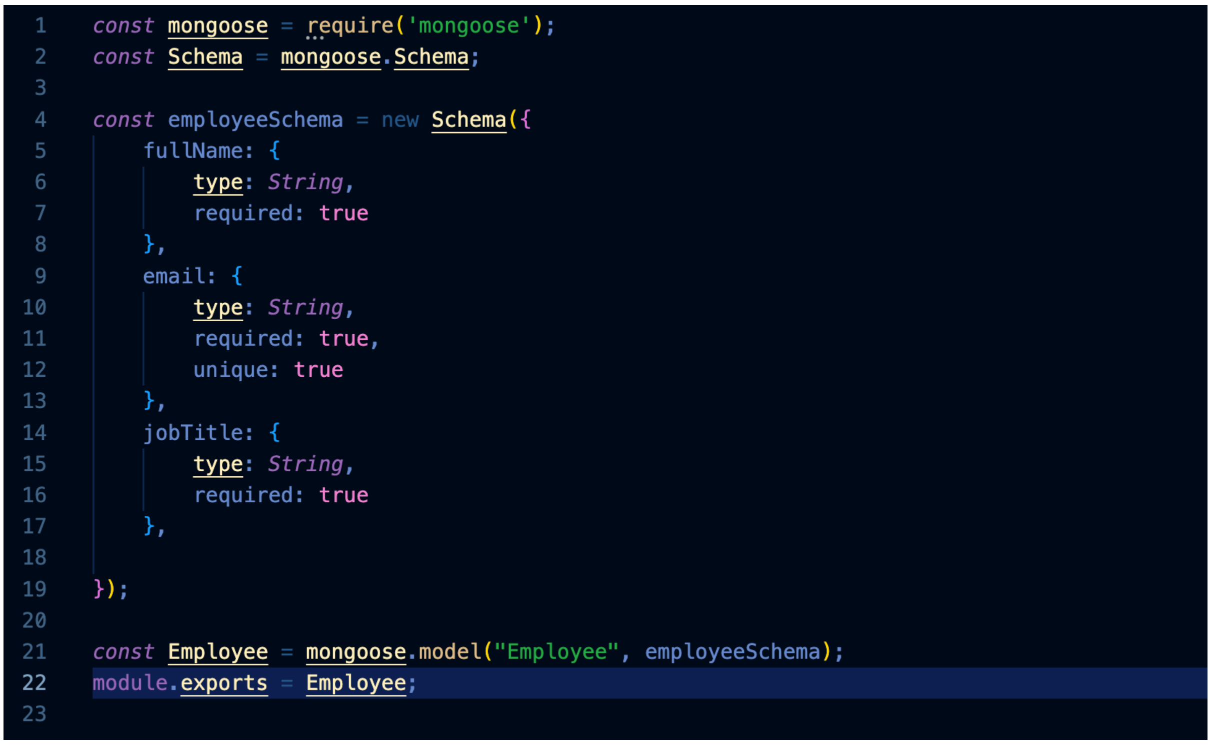
```
        IF job role is valid THEN
            CALCULATE salary based on role and hours worked
            STORE calculated salary for the employee
        ELSE
            DISPLAY "Invalid job role for employee"
        END IF
    END FOR
    DISPLAY "Salaries calculated successfully"

    RETURN to Step 4

Step 6: Generate Salary Report
    IF salaries are calculated THEN
        DISPLAY salary report for all employees
    ELSE
        DISPLAY "No salary data available"

Step 7: Create Petty Cash Form
    GET petty cash amount
    GET purpose of expense
    VALIDATE expense details
    IF details are valid THEN
        CREATE petty cash form and store details
        DISPLAY "Petty cash form created successfully"
    ELSE
        DISPLAY "Invalid petty cash details"
    END IF

    RETURN to Step 4
```

```
Step 8: Create Balance Sheet
    GET income and expenses data
    VALIDATE data
    IF data is valid THEN
        CALCULATE total income and total expenses
        GENERATE balance sheet
        DISPLAY "Balance sheet created successfully"
    ELSE
        DISPLAY "Invalid income or expenses data"
    END IF

    RETURN to Step 4

Step 12: Logout if necessary
    DISPLAY "Logging out..."
    END application

END
```

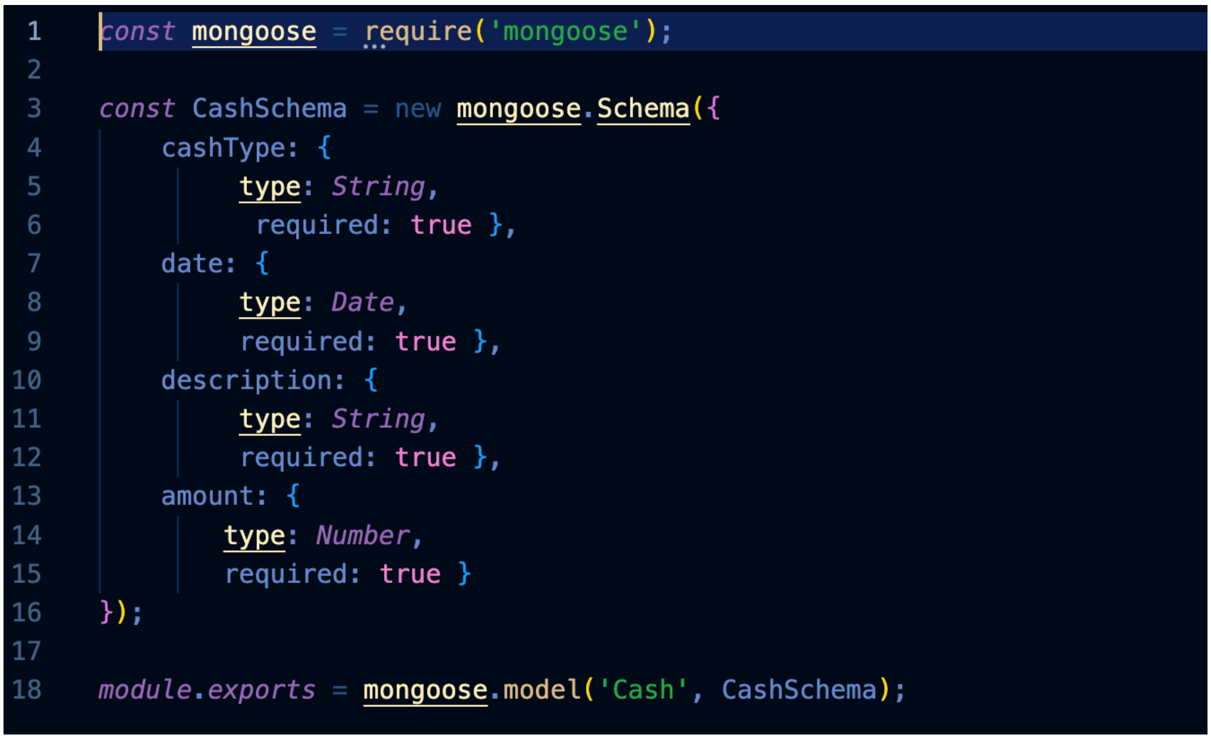
Data Base Schemas



```
const mongoose = require('mongoose');

const SalarySchema = new mongoose.Schema({

    userId: {

        type: mongoose.Schema.Types.ObjectId,
        ref: 'Employee',
        required: true },

    BasicSalary: {
        type: Number,
        required: true },
    Bonus: {
        type: Number,
         required: true },
    OTHours: {
        type: Number,
        required: true },
    OTRate: {
        type: Number,
        required: true },
    OTAmount: {
        type: Number,
        required: true },
    EPF: {
        type: Number,
         required: true },
    ETF: {
        type: Number,
        required: true },
    TotalSalary: {
        type: Number,
        required: true }
});

module.exports = mongoose.model('Salary', SalarySchema);
```



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const employeeSchema = new Schema({
    fullName: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    jobTitle: {
        type: String,
        required: true
    },

});

const Employee = mongoose.model("Employee", employeeSchema);
module.exports = Employee;
```



```
const mongoose = require('mongoose');

const CashSchema = new mongoose.Schema({
    cashType: {
        type: String,
         required: true },
    date: {
        type: Date,
        required: true },
    description: {
        type: String,
        required: true },
    amount: {
       type: Number,
       required: true }
});

module.exports = mongoose.model('Cash', CashSchema);
```

Auction Event Handling (IT22235138 Thalangama A. I)

Seating Management Flow Coordination – The system coordinates the assignment of seats to bidders, updates the seating plan in real-time, and sends email notifications to bidders about their seat numbers. The system should ensure accurate display and updates for seating plans according to auction locations.

Notification System – Ensure that emails sent to bidders are timely and contain the correct information (seat number, location). Use a reliable email service (like SendGrid, Nodemailer) to manage email notifications efficiently.

Seating Plan Integration – The seating plan needs to be displayed and updated dynamically as seats are assigned or unassigned. The system should visually differentiate between booked and available seats for various auction locations.

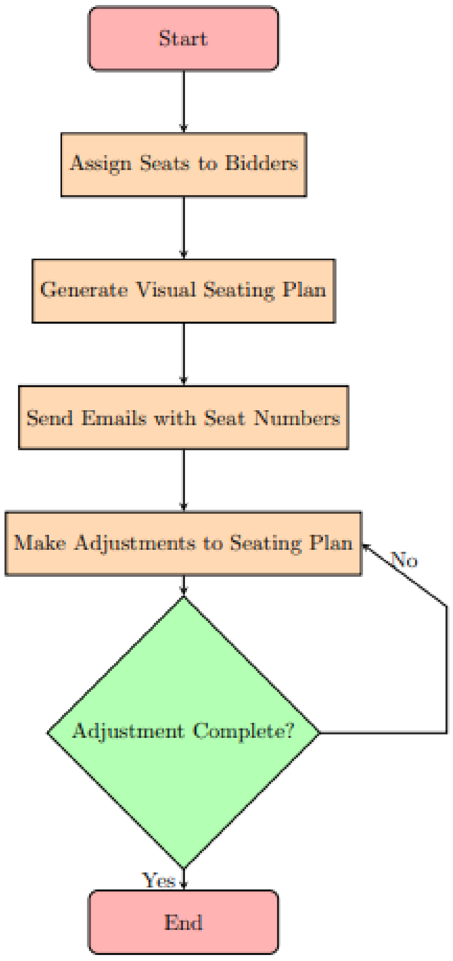
Seating Plan Updates – The seating plan must reflect seat status changes (booked, available) for all locations when seats are assigned or unassigned. This ensures that the auction organizers and bidders have an up-to-date view of seating arrangements

Work Progression

Completed Work	Work in Progress	Work to be Completed
Implemented seat assignment functionality where the seating manager manually assigns seats to bidders.	Developing the email notification system to inform bidders of seat assignments..	Implement automatic seat assignment to distribute available seats based on the number of unassigned bidders.
Updated the seating plan to mark seats as booked or available according to auction locations.	Working on seating plan updates across different auction locations (BMICH, Negombo Regal, Joash Place Maharagama).	Implement validations to ensure seat uniqueness within each location and prevent double bookings.

Displayed seating plan for different auction locations using a visual layout (color-coded booked and available seats).	Testing the full functionality of the seat unassignment process and ensuring that it reflects correctly in the seating plan.	Test and validate the system's behavior for different seat assignment/unassignment scenarios.

flow chart

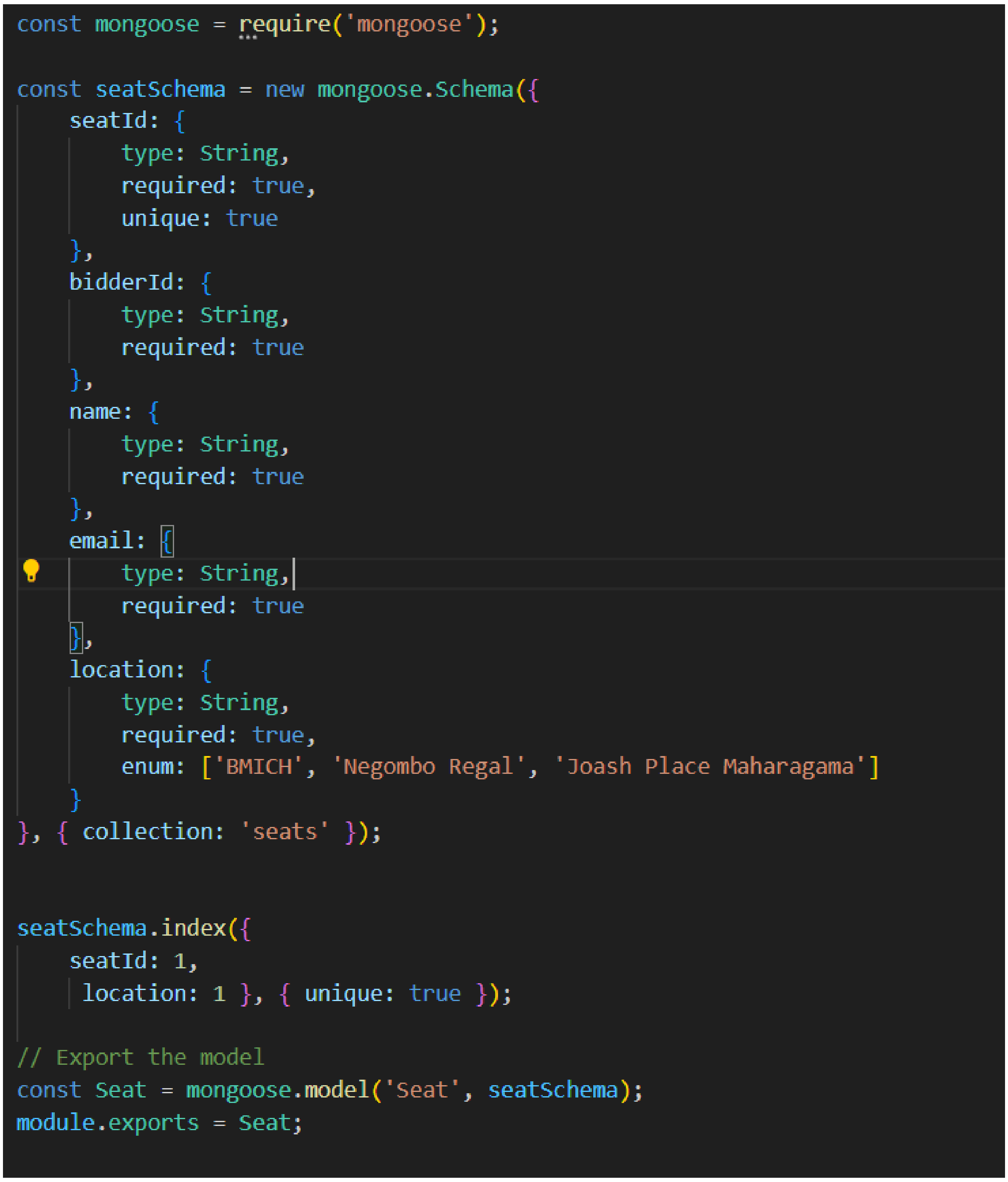

Start
Assign Seats to Bidders
Generate Visual Seating Plan
Send Emails with Seat Numbers
Make Adjustments to Seating Plan
No
Adjustment Complete?
Yes
End

Algorithm: Auction Event Handling

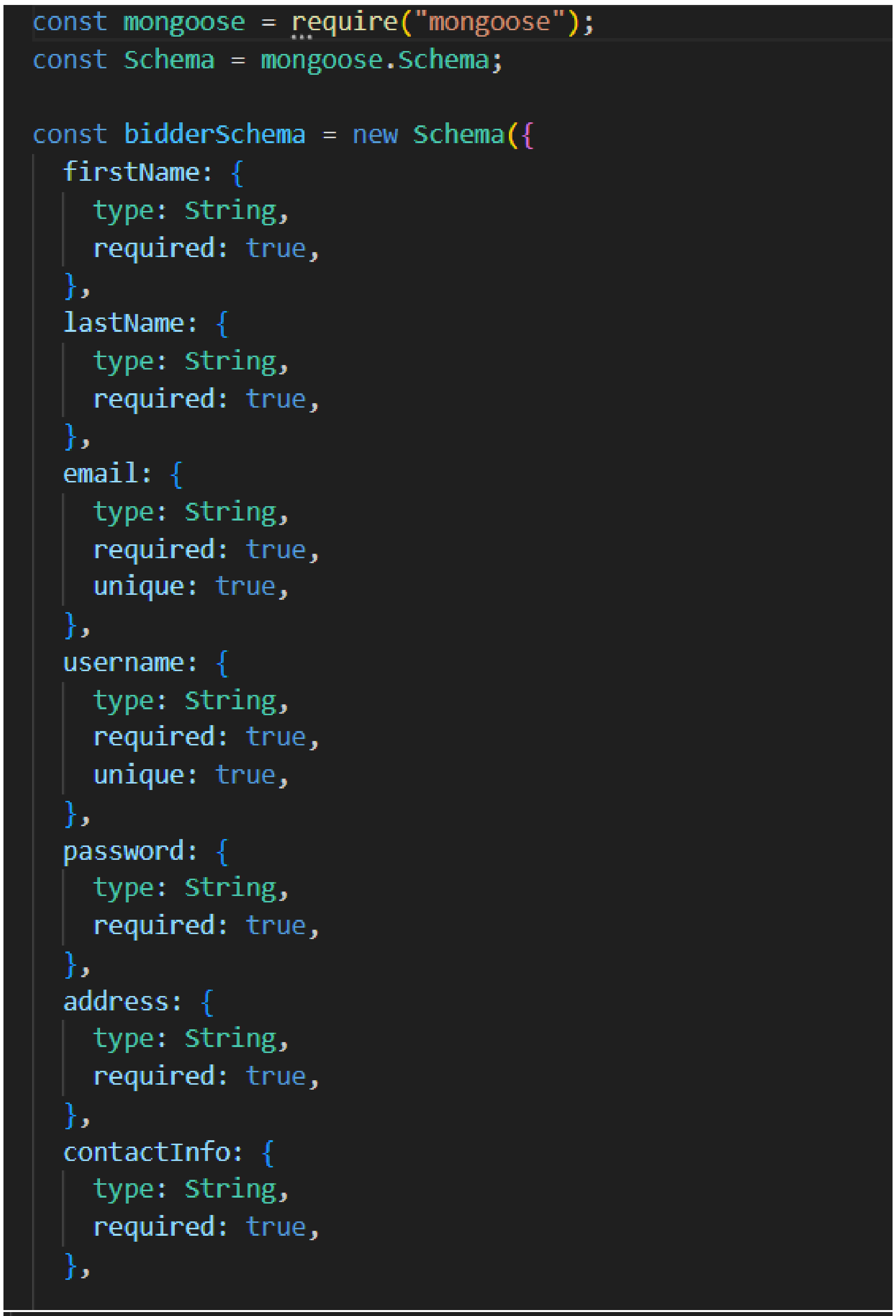
1. **Start**
2. **Access Auction Management System**
3. **Login to the system**
    - If credentials are incorrect: Display error message "Invalid credentials" and retry login.
    - If correct credentials:
        - If user is a seating manager: Proceed to Step 4.
        - If user is not a seating manager: Display "Unauthorized access" and return to Step 3.
4. **Seating Manager's Dashboard**:
    - Display the list of registered bidders and seating plan by auction locations.
    - **Select "Assign Seat"**:
        - Choose a bidder and auction location.
        - Select an available seat.
        - Assign the seat to the selected bidder.
        - **Update** the seating plan to mark the seat as booked.
        - **Send email** to the bidder with seat number and location.
        - If successful, display "Seat assigned successfully."
        - If errors occur, display error message and retry.
5. **Select "Unassign Seat"**:
    - Choose a bidder and seat details.
    - Unassign the bidder from the seat.
    - **Update** the seating plan to mark the seat as available.
    - If successful, display "Seat unassigned successfully."
    - If errors occur, display error message and retry.
6. **Select "View Seating Plan"**:
    - Choose an auction location.
    - **Display** the seating plan with color codes for booked and available seats.
7. **Logout (optional)**
8. **End**

```
BEGIN


Step 1: Start the application


Step 2: Access the Auction Management System


Step 3: Login to the system
   IF credentials are incorrect THEN
      DISPLAY "Entered username and password don't match"
      RETURN to Step 3
   ELSE
      IF user is a seating manager THEN
         GO TO Step 4
      ELSE
         DISPLAY "Unauthorized access"
         RETURN to Step 3
      END IF
   END IF


Step 4: Display seating manager's dashboard
   DISPLAY list of registered bidders and seating plan by auction locations


Step 5: Manage seat assignment
   IF "Assign Seat" is selected THEN
      SELECT a bidder and auction location
      SELECT available seat from the seating plan
      ASSIGN the seat to the selected bidder
      UPDATE the seating plan to mark the seat as booked
      SEND email to the bidder with seat number and location
      DISPLAY confirmation message "Seat assigned successfully"
   END IF
```

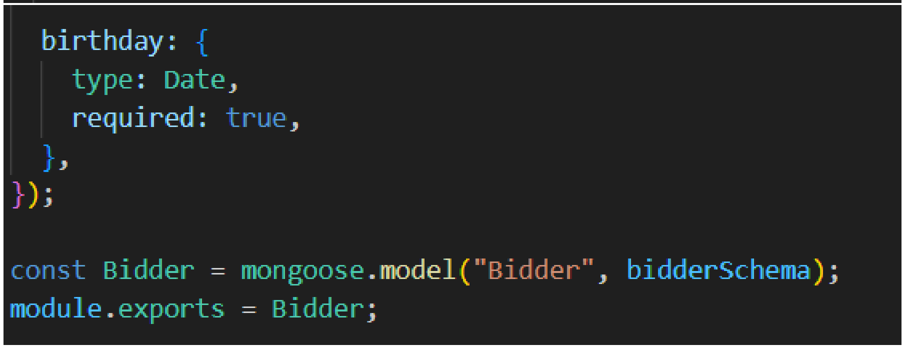
```
Step 6: Manage seat unassignment
  IF "Unassign Seat" is selected THEN
    SELECT the seat and bidder details
    REMOVE the bidder details from the seat
    UPDATE the seating plan to mark the seat as available
    DISPLAY confirmation message "Seat unassigned successfully"
  END IF

Step 7: Display seating plan
  IF "View Seating Plan" is selected THEN
    DISPLAY seating plan for the selected auction location
    INDICATE booked and available seats using color codes
  END IF

Step 8: Automatic seat assignment (if applicable)
  IF "Auto Assign Seats" is selected THEN
    FETCH all unassigned bidders and available seats
    AUTOMATICALLY assign seats based on availability and location preferences
    UPDATE seating plans accordingly
    SEND seat assignment emails to all assigned bidders
    DISPLAY confirmation message "Seats assigned successfully"
  END IF

Step 9: Logout if necessary

END
```



```javascript
const mongoose = require('mongoose');

const seatSchema = new mongoose.Schema({
    seatId: {
        type: String,
        required: true,
        unique: true
    },
    bidderId: {
        type: String,
        required: true
    },
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    location: {
        type: String,
        required: true,
        enum: ['BMICH', 'Negombo Regal', 'Joash Place Maharagama']
    }
}, { collection: 'seats' });


seatSchema.index({
    seatId: 1,
     location: 1 }, { unique: true });

// Export the model
const Seat = mongoose.model('Seat', seatSchema);
module.exports = Seat;
```



```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const bidderSchema = new Schema({
  firstName: {
    type: String,
    required: true,
  },
  lastName: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  username: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  address: {
    type: String,
    required: true,
  },
  contactInfo: {
    type: String,
    required: true,
  },
```



```
  birthday: {
    type: Date,
    required: true,
  },
});

const Bidder = mongoose.model("Bidder", bidderSchema);
module.exports = Bidder;
```