Test Document

for

BILLING 360

Version 1.00

Prepared by

Group 7: **Group Name: Team Arjuna**

Name	Roll No.	Email
Abhishek Khandelwal	220040	abhishekkh22@iitk.ac.in
Pallav Goyal	220747	pallavg22@iitk.ac.in
Kundan Kumar	220568	kundank22@iitk.ac.in
Saagar K V	220927	saagar22@iitk.ac.in
Dhruv Gupta	220361	dhruvgupta22@iitk.ac.in
Pragati Agrawal	220779	apragati22@iitk.ac.in
Poojal Katiyar	220770	poojalk22@iitk.ac.in
Ansh Agarwal	220165	ansha22@iitk.ac.in
Nipun Nohria	220717	nipun22@iitk.ac.in
Venkatesh Akula	220109	akulav22@iitk.ac.in

Course: CS253

Mentor TA: Somesh

Date: 1st April, 2024

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.00	Abhishek Khandelwal Ansh Agarwal Dhruv Gupta Kundan Kumar Nipun Nohria Pallav Goyal Poojal Katiyar Pragati Agrawal Saagar K V Venkatesh Akula	First version of the testing document	01/04/2024

1. Introduction

Test Strategy:

We used manual testing to test our software.

Testing period:

The majority of the testing was done after the implementation was completed. However, even during the implementation phase, we frequently did some manual testing of our code.

The Testers:

Developers were themselves the testers. However, we ensured that the person who wrote a particular functionality has not tested the same functionality.

Coverage Criteria:

We have used functional and non-functional coverage.

Tools used for testing:

We utilized **Thunder Client** to conduct comprehensive testing of our software. Thunder Client stands out as a *premier HTTP client extension* designed for Visual Studio Code, empowering developers to seamlessly test APIs and web services directly within their code editor environment. Its key advantages include:
Intuitive Interface: Thunder Client boasts an intuitive and user-friendly interface, streamlining the process of setting up and executing requests. Developers can swiftly navigate through functionalities without grappling with intricate configurations.
Seamless Integration: Thunder Client seamlessly integrates with Visual Studio Code, facilitating a smooth transition between coding and testing phases. This integration enhances workflow efficiency by eliminating the need to switch between disparate environments.
Robust Feature Set: Thunder Client offers a plethora of advanced features, including automated cookie management, comprehensive request history tracking, and readily available code snippets. These features empower developers to optimize productivity and minimize time spent on repetitive tasks.
Open Source: Thunder Client is built on open-source principles, providing developers with the freedom to utilize the tool without incurring any licensing fees. Moreover, its open nature encourages community contributions, fostering continuous improvement and innovation.

2. Unit Testing

Unit testing was mainly done using ThunderClient. Each individual unit was tested separately with special attention to inputs that may cause anomalous behaviour. Here, backend and frontend are tested as separate units. Their integration has been tested in integration testing. Testing of a backend unit includes the passing of input as a JSON object and performing the API call. Now, the obtained response is compared with the expected response. Testing of the frontend unit includes ensuring that inputs are converted into JSON objects correctly, all checks work as expected and alerts are displayed correctly.

We have performed the unit testing for each section exhaustively by using various test cases to model every possibility. For documentation, we have listed these sections. Each section deals with individual units which further contain many test cases. Repeated features like searching, sorting etc have been mentioned in only one of the sections, although they were tested in all of them.

1(a). SignUp Form

In this unit, we have tested the backend function that creates a new user.

Unit Details: [User class, registerUser() function]. Testing of this unit ensures that a new user is registered correctly with the entered details. It also ensures that no two users are registered with the same email.

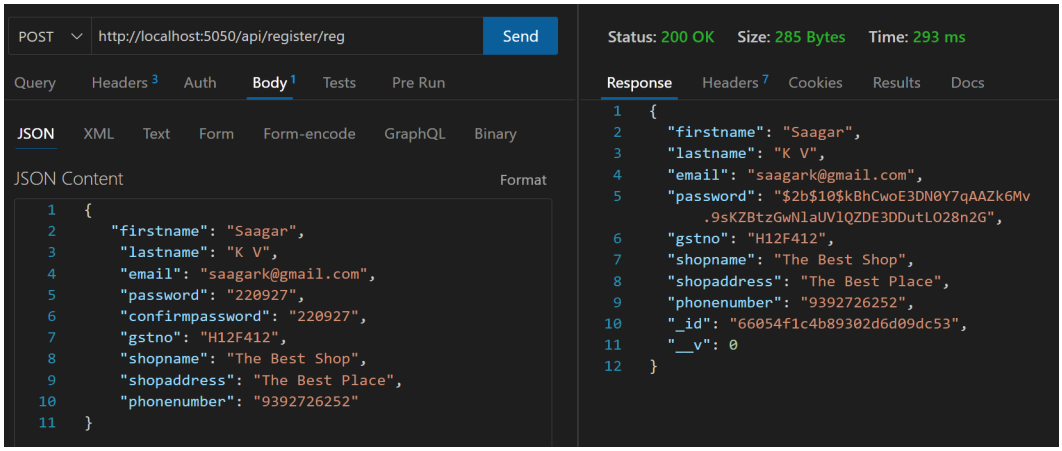
Test Owner: Saagar K V

Test Date: 26/03/2024

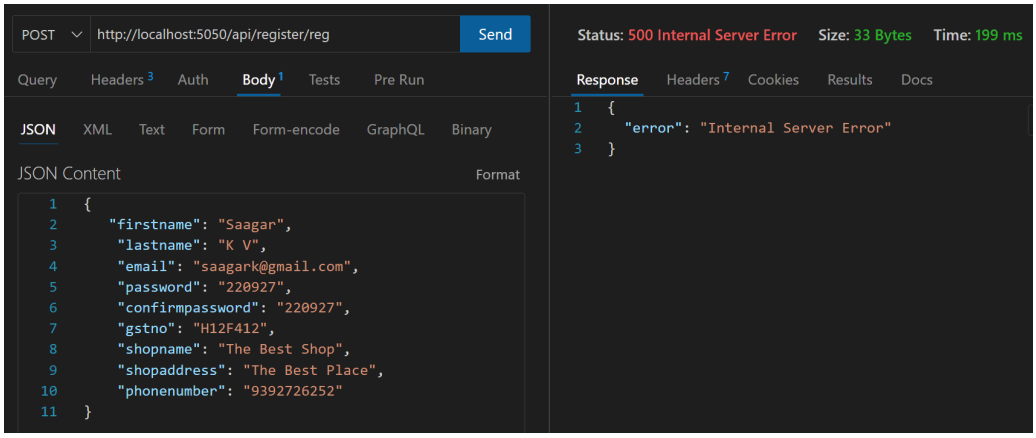
Test Results: A new user was correctly registered. An object of the ‘User’ class (a new document in the ‘users’ collection in the database) was created with correct details. When registration was attempted with the same email again, it was disallowed by the system as expected.

Structural Coverage: Functional coverage (The controller registerUser()), Decision (and Branch) coverage (covers the check for existing user with the same email).

i) New Email:



ii) Existing email:



1(b). **Sign Up form**

In this unit, we have tested the frontend associated with the sign up form.

Unit Details: [User class, Register() function]. This unit ensures that a required field is not missing. It ensures other requirements such as minimum length of password, valid email format, agreement to terms and conditions and equality of entry in password and confirm password fields. It also confirms that details are correctly converted to JSON object to be sent in the request to the backend.

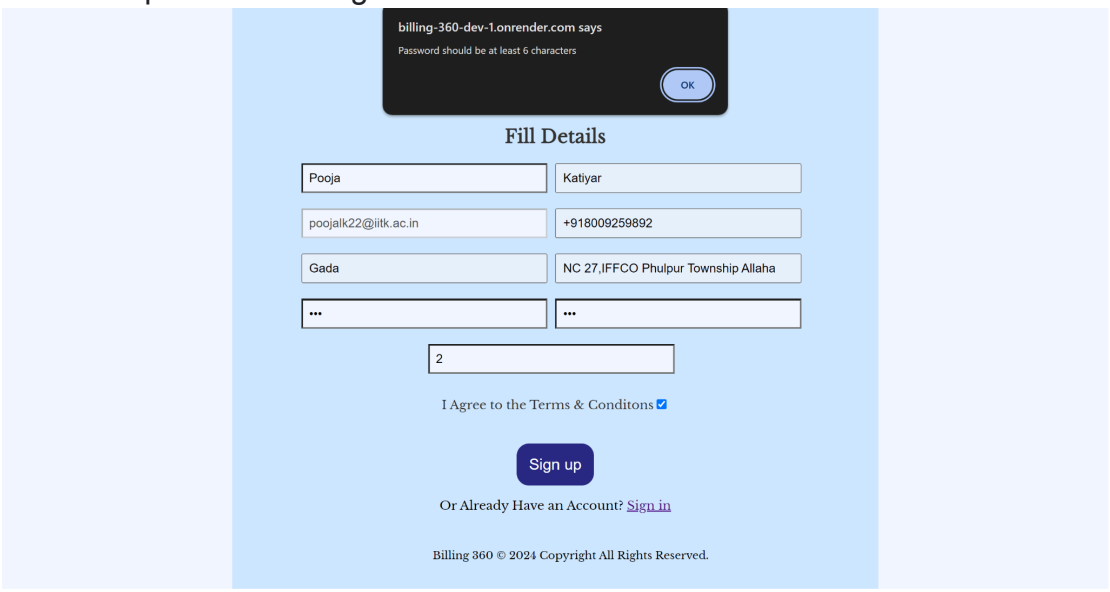
Test Owner: Venkatesh Akula

Test Date: 26/03/2024

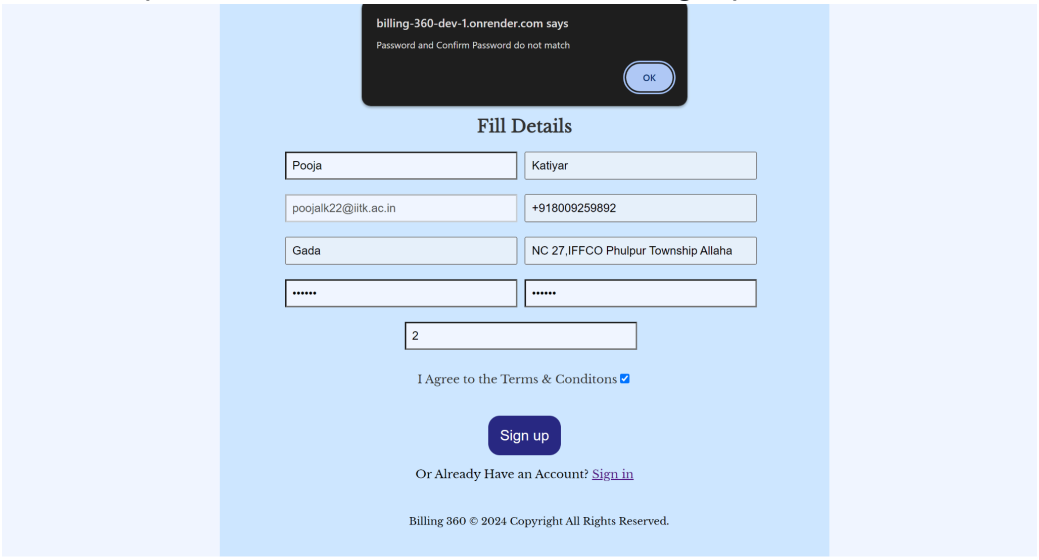
Test Results: Alerts were shown if an entry is invalid or if a required field is missing. The details were correctly converted to a JSON object.

Structural Coverage: Covers the code in Register.js file, Branch coverage (covers the check for valid format of credentials and missing credentials).

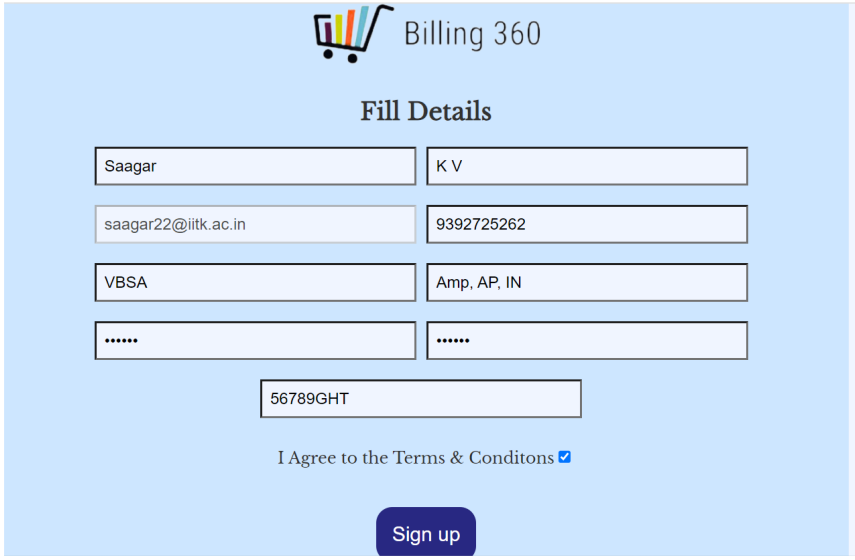
Check for password being at least 6 characters:



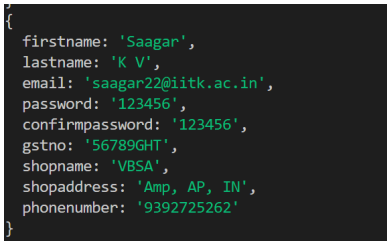
Check for password and confirm Password being equal:



Upon entering valid details, the JSON object is correctly created to be sent in the request to the backend:



(Entered details in SignUp form)



(JSON object logged onto the console)

2(a). **Login Form**

In this unit, we have tested the backend functions that control authentication.

Unit Details: [User class, loginUser() function]. Testing of this unit ensures that a valid user is allowed to sign in, if the entered details are correct. It also ensures that an unauthenticated user is not allowed to sign in.

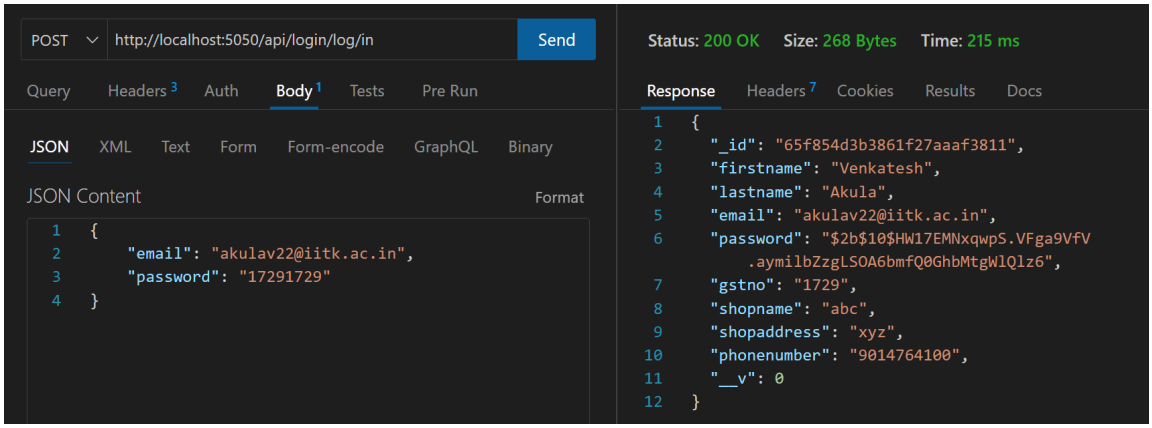
Test Owner: Venkatesh Akula

Test Date: 26/03/2024

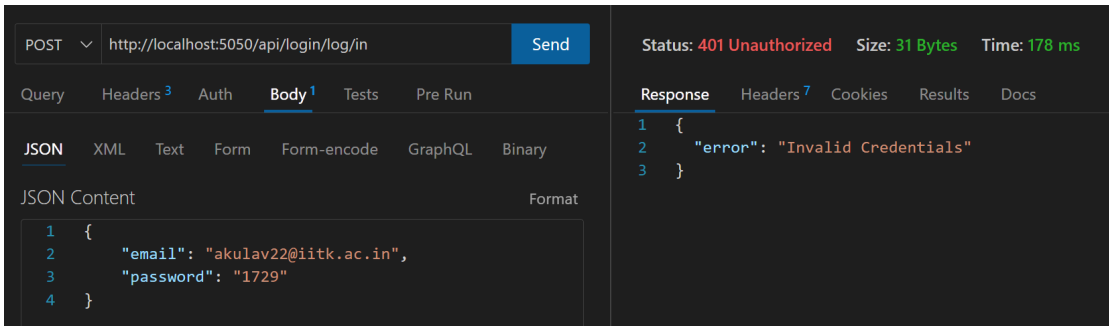
Test Results: An authenticated user (valid details) was allowed to log in successfully. An unauthenticated user was prevented from logging in.

Structural Coverage: Functional coverage (covers the loginUser() and getUserData() functions), Branch coverage (covers the check for valid credentials).

i) Valid credentials:



ii) Invalid credentials:



2(b). **Login Form**

In this unit, we have tested the frontend associated with the login form.

Unit Details: [User class, Login() function]. This unit ensures that no field is missing and details are correctly converted to JSON object to be sent in the request to the backend.

Test Owner: Venkatesh Akula

Test Date: 26/03/2024

Test Results: Alerts were shown if a field is missing. The details were correctly converted to a JSON object.

Structural Coverage: Covers the code in Login.js file, Branch coverage (covers the check for missing credentials).

Missing credential:

3. Dashboard

This unit deals with the backend that controls the fetching of data and calculation of information to be displayed on the dashboard.

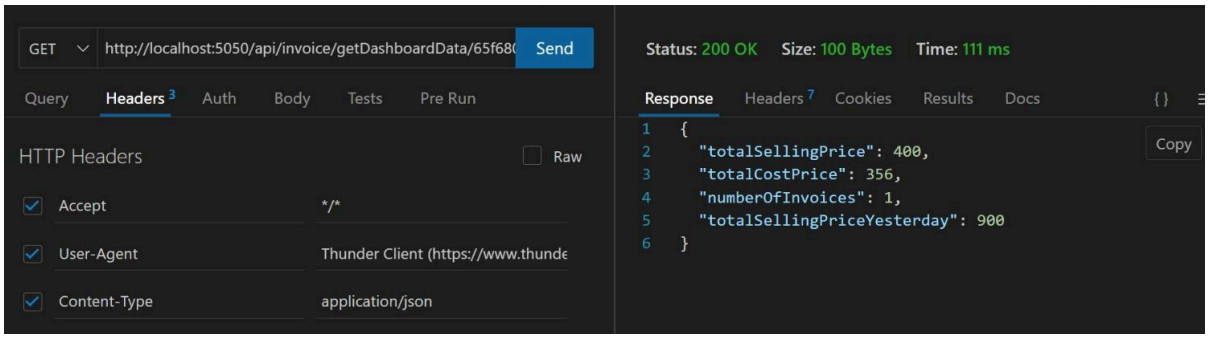
Unit Details: [Transaction class, getDashboardData() function]

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: Data obtained was correct.

Structural Coverage: Functional coverage(It covers getDashboardData() function).



4. **Invoice**

a) (i) **Add new invoice** (backend testing)

In this unit, we have tested whether a new invoice gets generated correctly (backend testing).

Unit Details: [Invoice class, addInvoice() function].

Test Owner: Saagar K V

Test Date: 26/03/2024

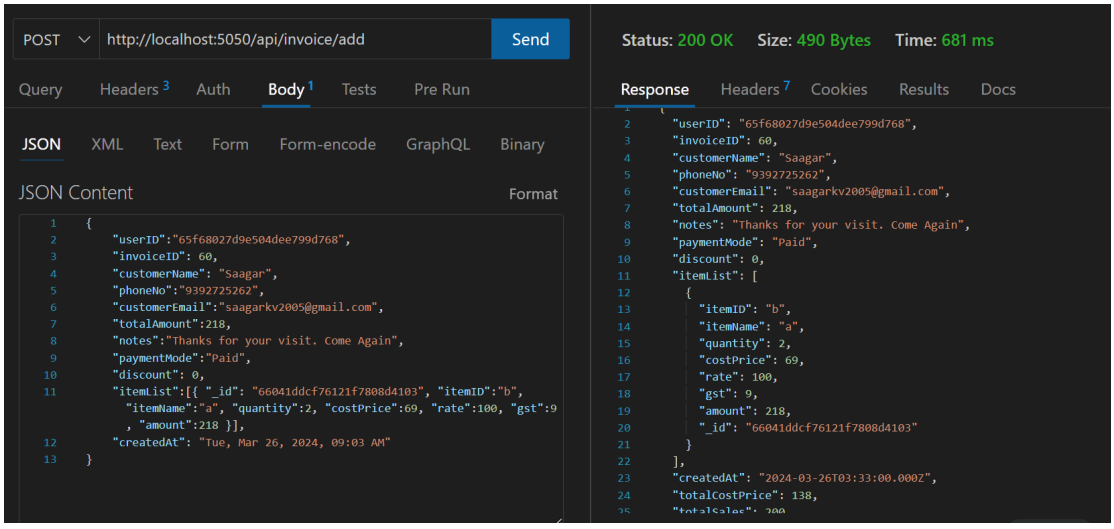
Test Results: An invoice was correctly generated. A new invoice object (a new document in the ‘invoices’ collection) was created in the database.

Structural Coverage: Functional coverage (covers the addInvoice() function), Branch coverage (covers the two possibilities which arise depending on whether the customer is new or already existing).

Two different test cases are required for branch coverage:

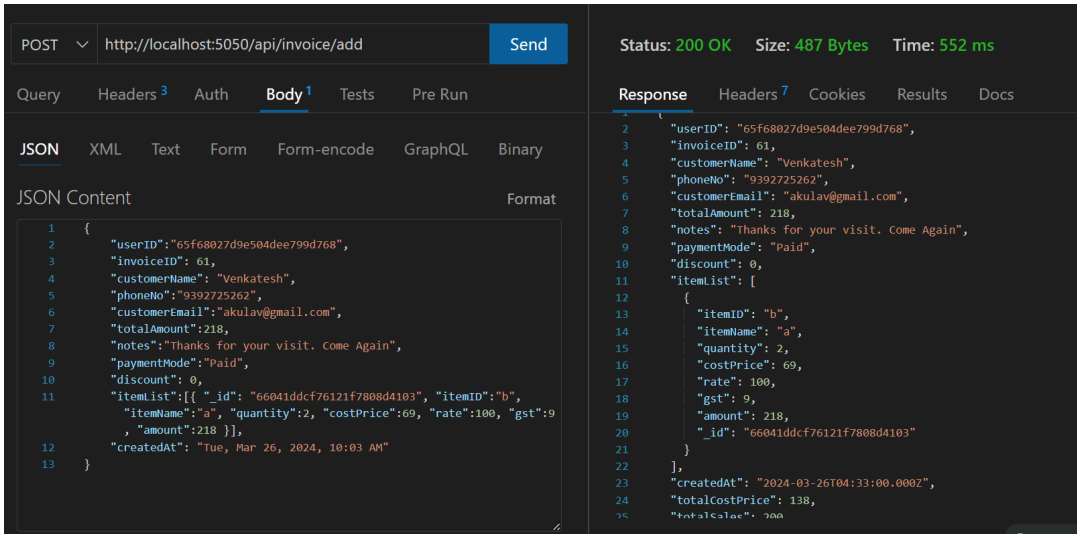
i) Existing Customer :

Invoice was correctly generated and added to the customer’s invoice list.



ii) New Customer:

A new customer was created. The newly generated invoice was added to his invoice list.



(ii) Add new invoice (frontend testing)

In this unit, we have tested the frontend associated with the invoice generation page.

Unit Details: [User class, AddNewInvoice() function]. This unit ensures that no field is missing and details are correctly converted to a JSON object to be sent in the request to the backend. It also ensures that email and phone number are of the correct format. It prevents generation of empty bills as well.

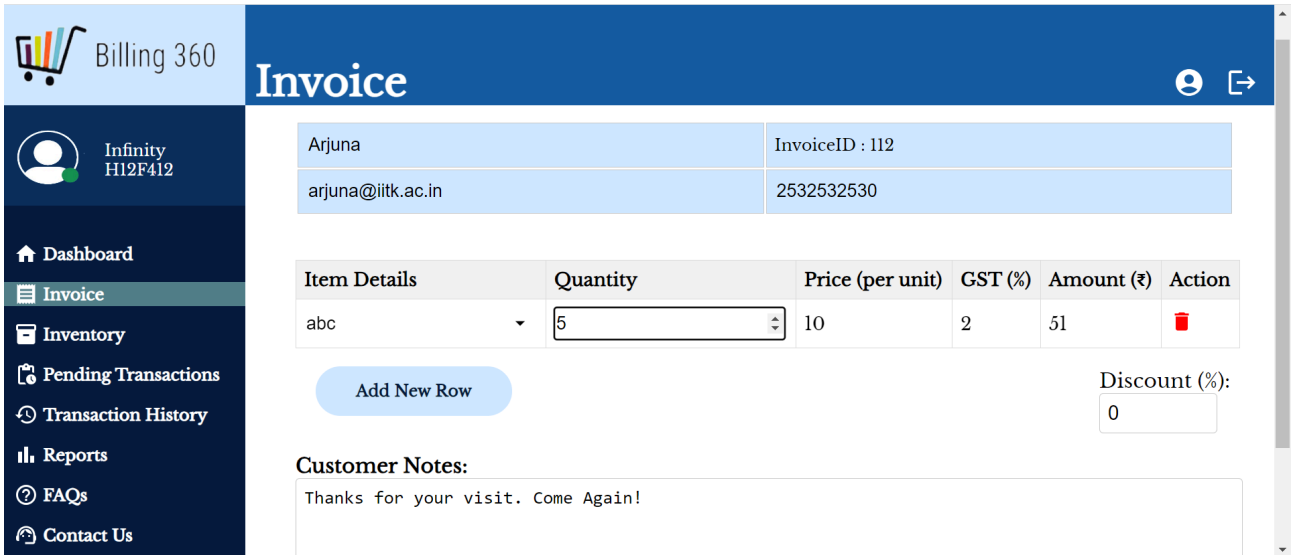
Test Owner: Venkatesh Akula

Test Date: 26/03/2024

Test Results: Valid details were correctly converted to a JSON object. In case of invalid details, appropriate alerts were shown.

Structural Coverage: Covers the code in AddInvoice.js file, Branch coverage (covers the checks for validity of various credentials).

Details entered while creating new invoice

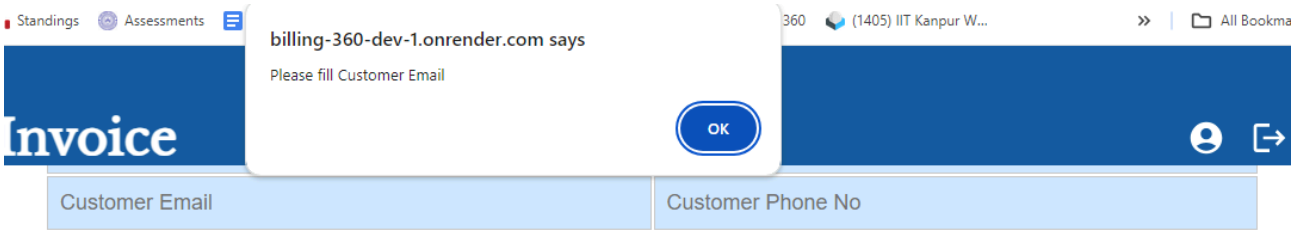


JSON object logged onto the console


```
req:  {
  userID: '65f68027d9e504dee799d768',
  invoiceID: 112,
  customerName: 'Arjuna',
  phoneNo: '2532532530',
  customerEmail: 'arjuna@iitk.ac.in',
  totalAmount: 51,
  notes: 'Thanks for your visit. Come Again!',
  paymentMode: 'Paid',
  discount: 0,
  itemList: [
    {
      _id: '6607e53df64c8bfe26cdbcfe',
      itemID: '',
      itemName: 'abc',
      quantity: 5,
      costPrice: 9,
      rate: 10,
      gst: 2,
      amount: 51
    }
  ],
  createdAt: '2024-04-01T21:20:08.212Z'
}
```

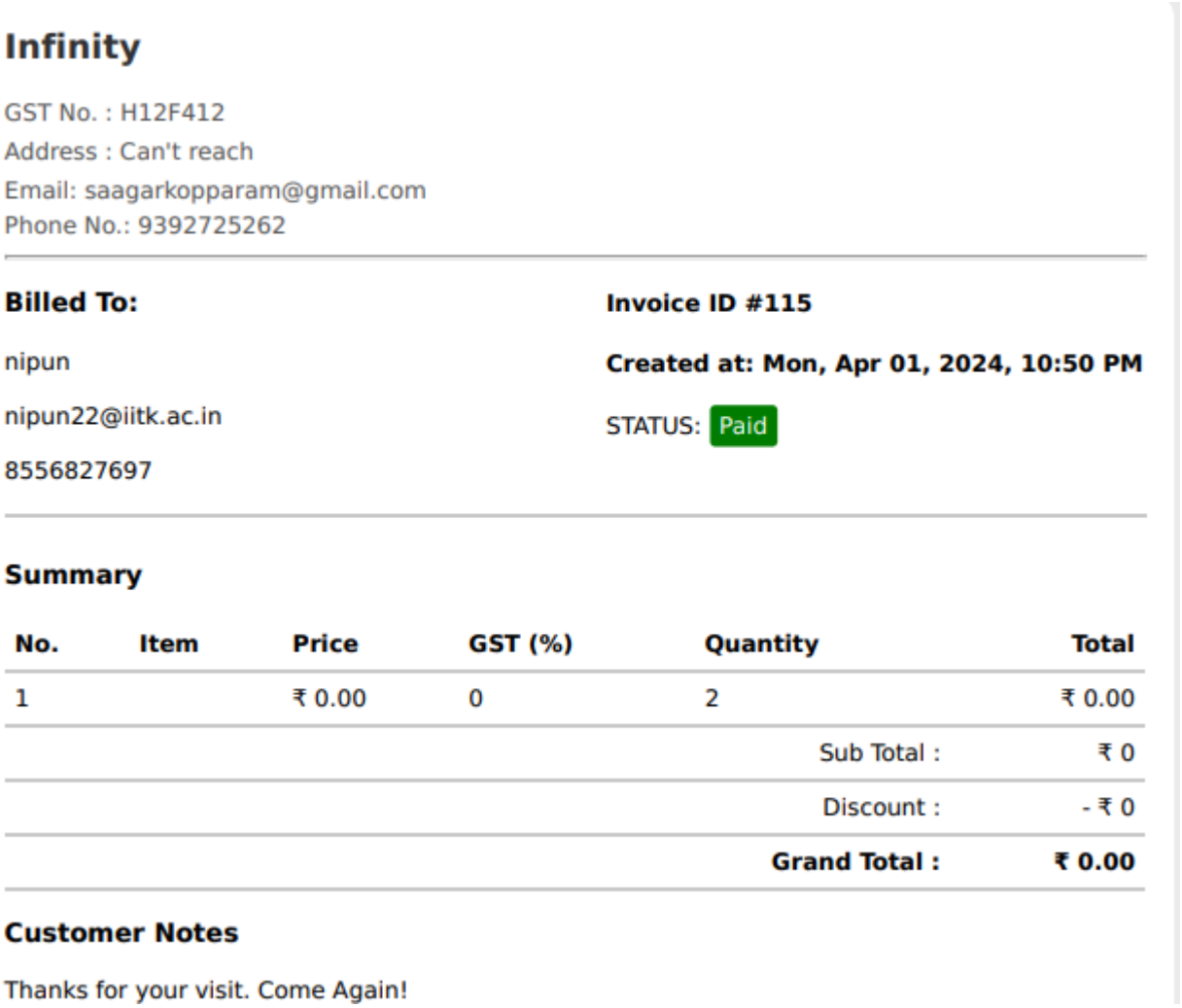

II)If customer details are not entered while generating bill.

Result:Invoice is not added and message is generated.



III)f product Name is not added

Result:No error is generated and invoice is still generated.



Infinity

GST No. : H12F412
Address : Can't reach
Email: saagarkopparam@gmail.com
Phone No.: 9392725262

Billed To:

nipun

nipun22@iitk.ac.in

8556827697

Invoice ID #115

Created at: Mon, Apr 01, 2024, 10:50 PM

STATUS: Paid

Summary

No.	Item	Price	GST (%)	Quantity	Total
1		₹ 0.00	0	2	₹ 0.00
				Sub Total :	₹ 0
				Discount :	- ₹ 0
				Grand Total :	**₹ 0.00**

Customer Notes

Thanks for your visit. Come Again!

IV)Negative quantity is inputted.

Result:Negative sign is not typed in quantity input space.

b) Counting number of invoices:

In this unit, we have tested whether the number of invoices generated by a particular user (shopkeeper) is counted correctly (backend testing)

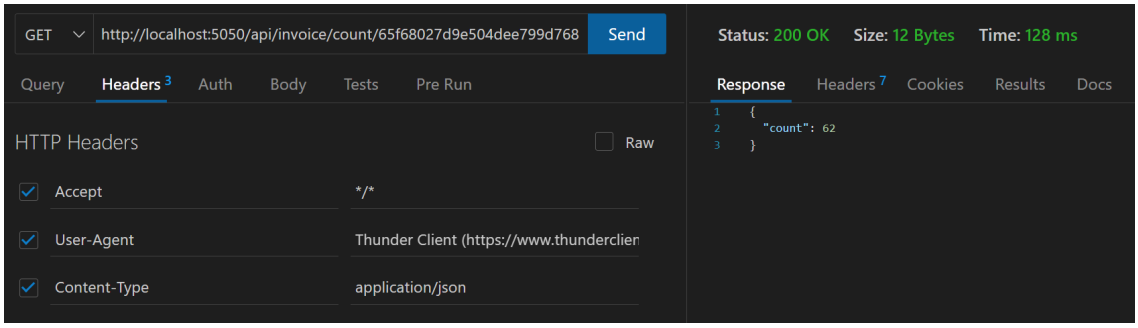
Unit Details: [Invoice class, getInvoiceCount() function]. Testing of this unit ensures that the counting of invoices of a particular user is correct. This is crucial since this count is used to generate invoice ID which must be unique for each invoice.

Test Owner: Saagar K V

Test Date: 27/03/2024

Test Results: The number of invoices returned was correct.

Structural Coverage: Functional coverage (covers the getInvoiceCount() controller function)



c) Generating PDF of invoice:

In this unit, we have tested the generation of invoice in pdf format (backend testing).

Unit Details: [Invoice class, generatePdf() function].

Test Owner: Saagar K V

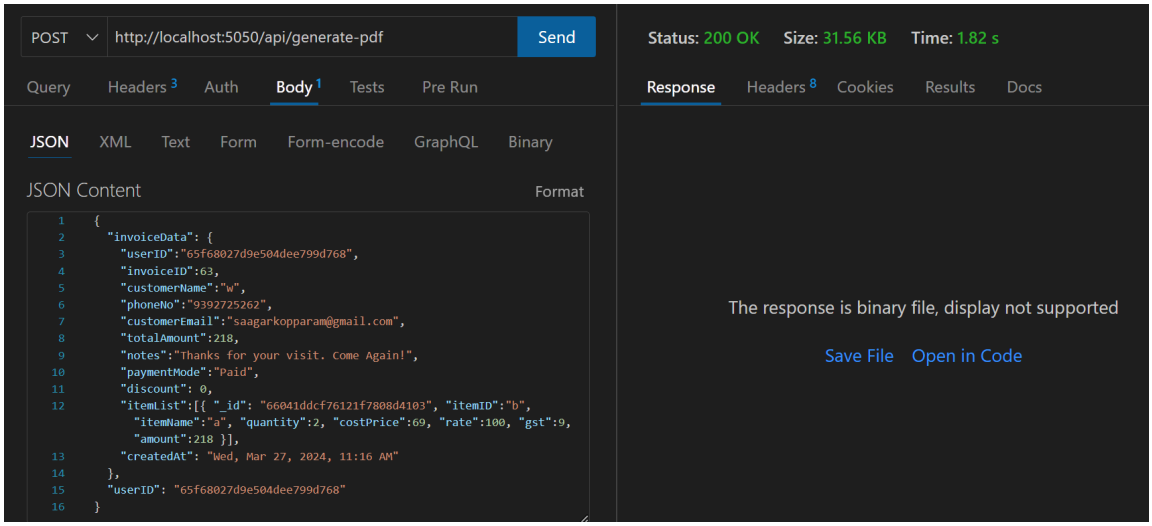
Test Date: 27/03/2024

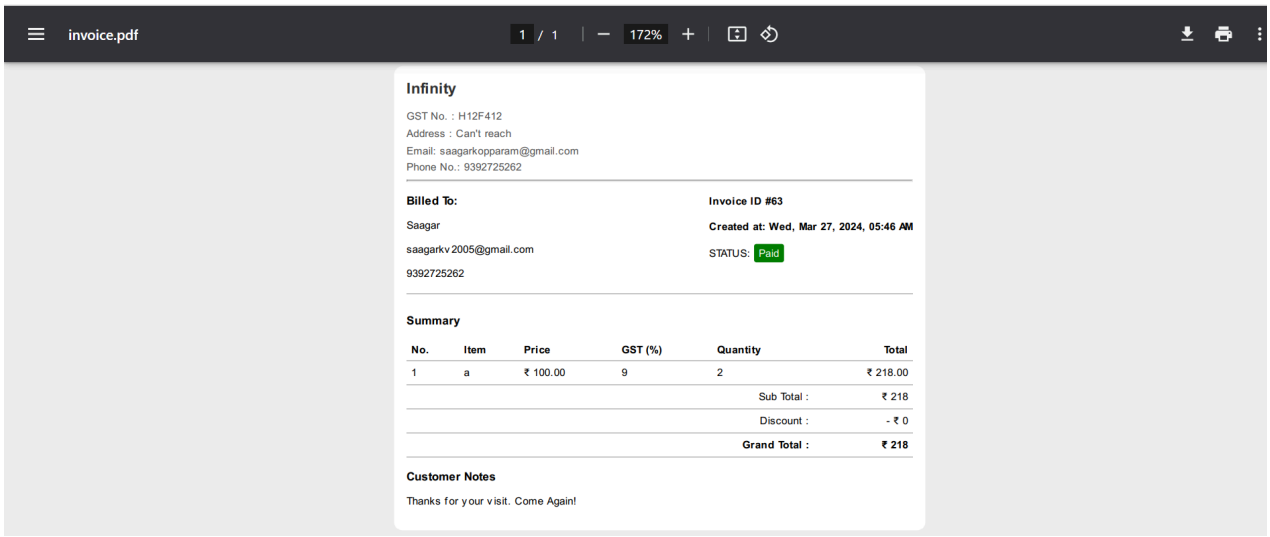
Test Results: The pdf was generated successfully.

Structural Coverage: Functional coverage (covers the generatePdfBuffer() function).

Additional comments: The time taken for generating the pdf was found to be variable. However, the time taken was found to lie in the range 1.5-2.5s, except during the first operation of the function when it takes 7-15s.

During testing, the time in the ‘createdAt’ field is given with respect to GMT, which is the case with the Date() function that has been actually used to obtain the current date and time in the function.





d) Sending the invoice to the customer via mail:

In this unit, we have tested the feature concerning the sending of mail to the customer (backend testing).

Unit Details: [Invoice class, sendInvoiceMail() function].

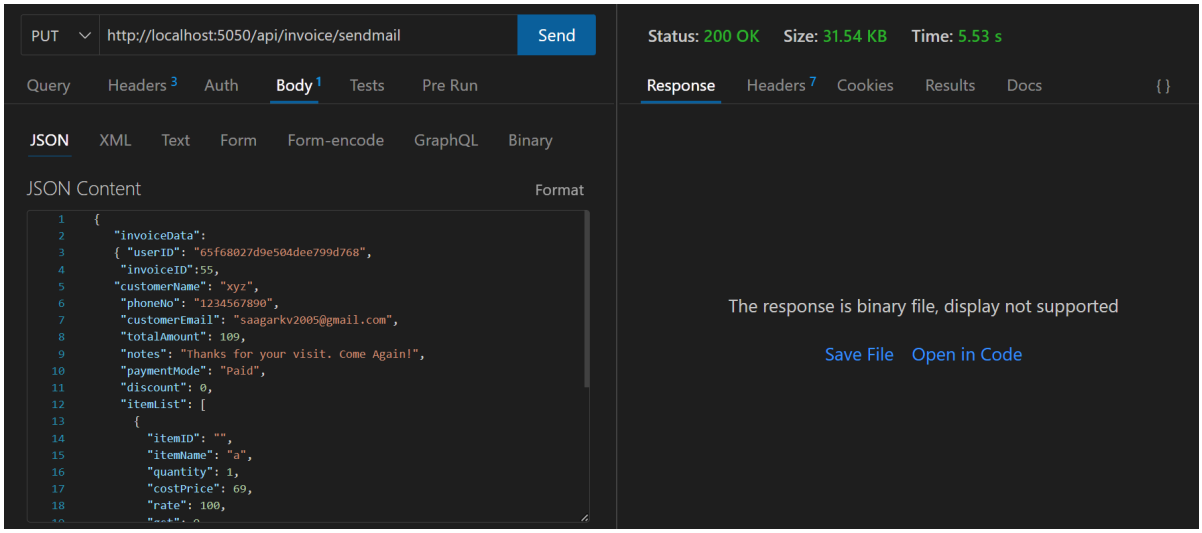
Test Owner: Saagar K V

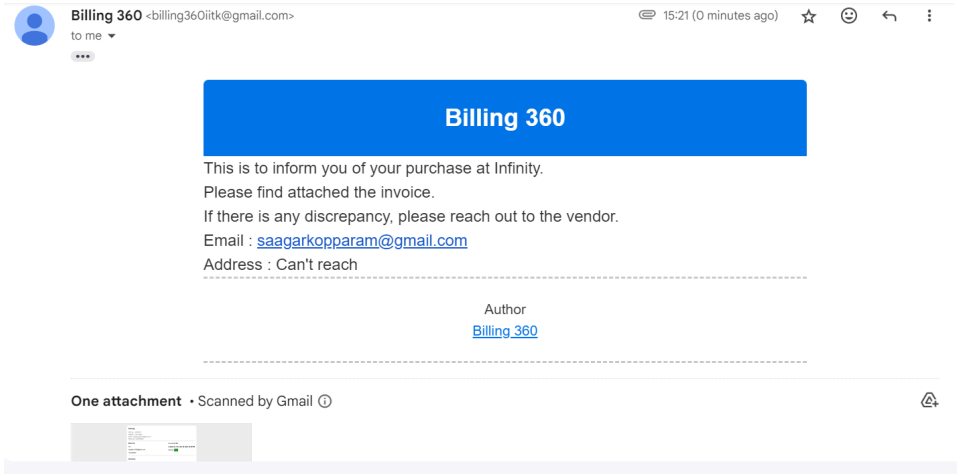
Test Date: 28/03/2024

Test Results: The mail was sent successfully with attached invoice, in pdf format.

Structural Coverage: Functional coverage (covers the generatePdfBuffer(), sendInvoiceMail() and sendInvoiceMailContoller() functions). It also confirms that 'nodemailer' works correctly.

Additional comments: The time taken for sending mail was found to be in the range 5-6s.





5. Pending transactions

a) Fetch all customers with non-zero credit

In this unit, we have tested whether all customers with non-zero credit are correctly fetched from the database (backend testing).

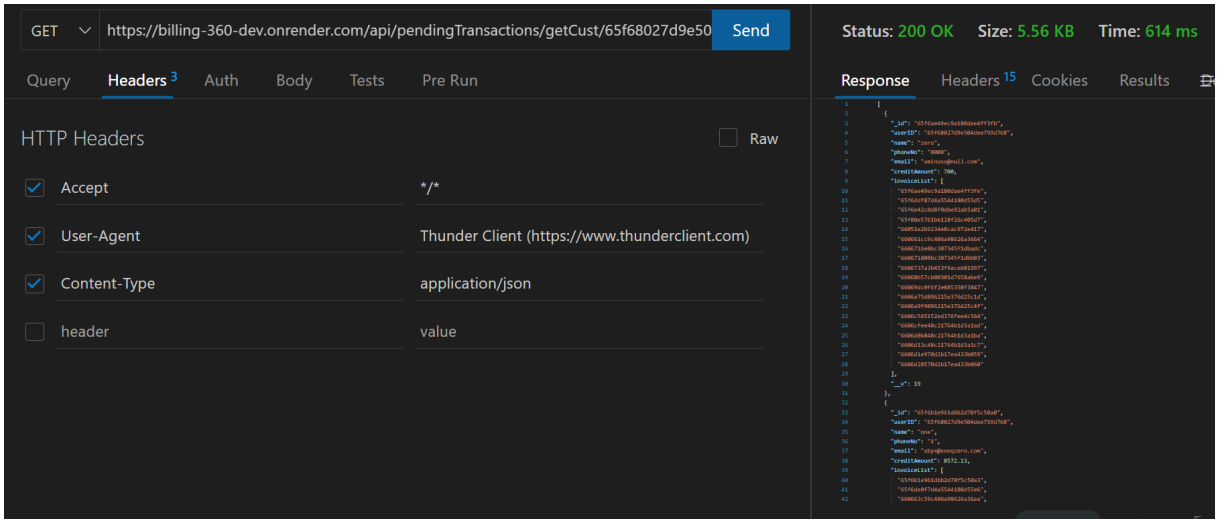
Unit Details: [Customer class, getCreditCustomers() function].

Test Owner: Saagar K V

Test Date: 28/03/2024

Test Results: The result obtained was correct.

Structural Coverage: Functional coverage (covers the getCreditCustomers() controller).



Frontend testing : All customers with non-zero credit amount were correctly displayed under the credit-tab. The total credit amount was also calculated correctly and displayed. Toggling of bars worked as expected. Under the debit tab, all suppliers with non-zero debit were displayed and the correct total debit amount was displayed. This covers the testing of fetchCreditCustomers(), fetchDebitSuppliers() and handleTotalAmt() functions in the frontend.

b) Find Existing Customer

In this unit, we have tested the function that returns a customer with a given email, if such a customer exists. Note that every customer is uniquely mapped (one-to-one) with an email.

Unit Details: [Customer class, existingCustEmail() function].

Test Owner: Saagar K V

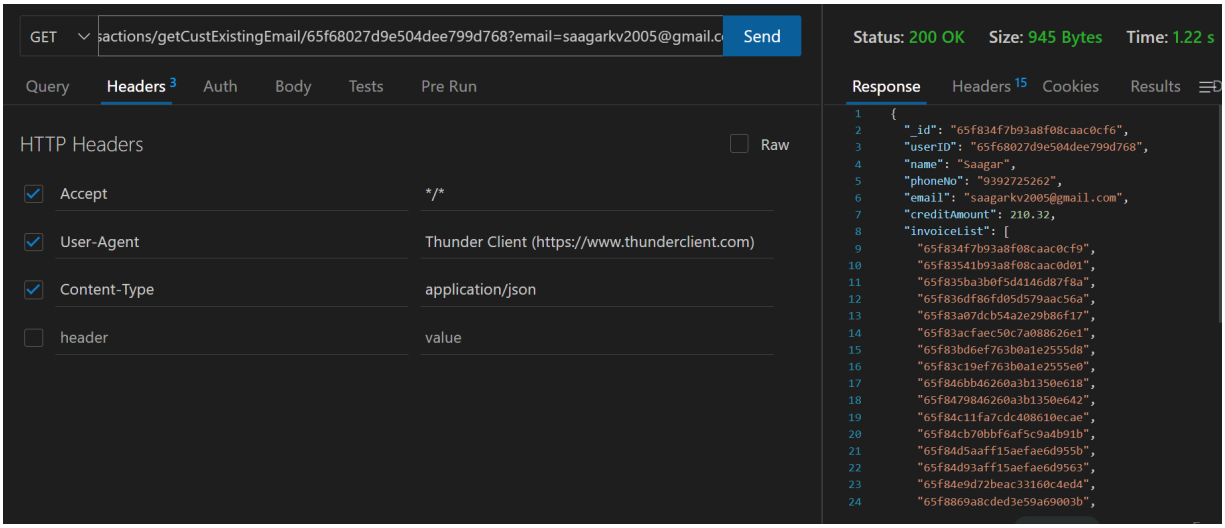
Test Date: 28/03/2024

Test Results: The result obtained was correct. If such a customer didn't exist, it returned null as expected.

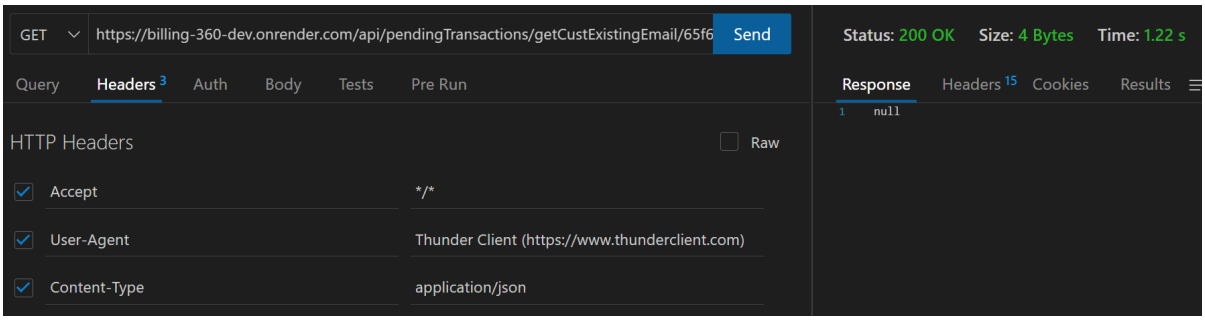
Structural Coverage: Functional coverage (covers the existingCustEmail() function).

Two separate test cases were used:

i) There is an existing customer with that email



ii) There is no such customer



c) (i) Add New Customer (backend testing)

In this unit, we have tested the addition of a new customer into the database (backend testing).

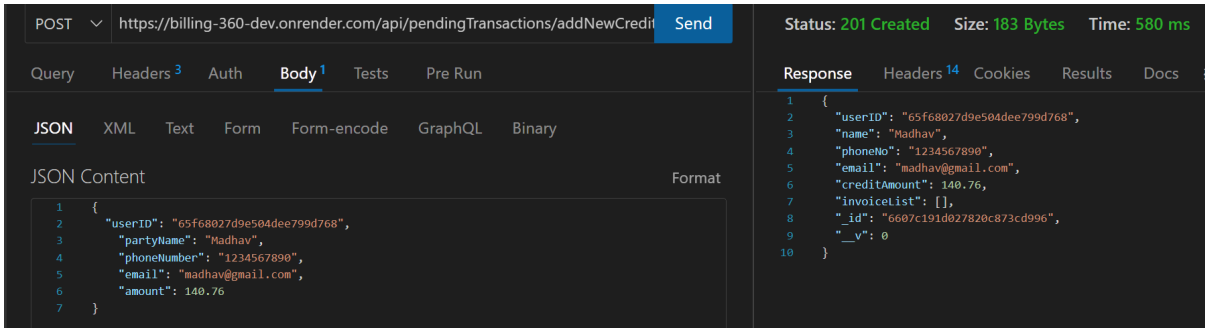
Unit Details: [Customer class, AddNewCredit() function].

Test Owner: Saagar K V

Test Date: 28/03/2024

Test Results: The customer was correctly added. A new document was successfully created in the ‘customers’ collection.

Structural Coverage: Functional coverage (covers the AddNewCredit() controller function)



(ii) Add New Customer (frontend testing)

In this unit, we have tested the addition of a new customer into the database (frontend testing).

Unit Details: [Customer class, AddNewEntry() function].

Test Owner: Saagar K V

Test Date: 28/03/2024

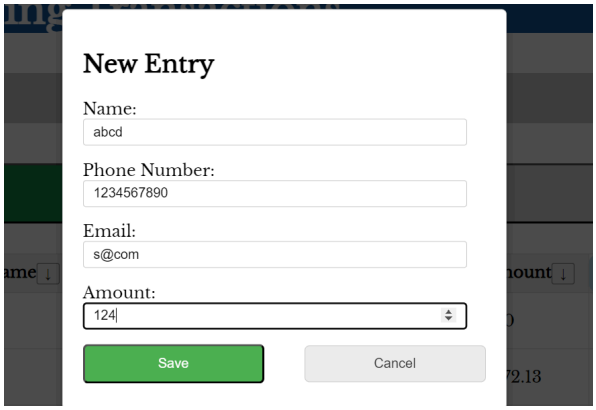
Test Results: If entered details are valid, it was correctly converted to a JSON object for sending the request to the backend. It checked if the phone number and email are of a valid format. It also required the amount to be a positive number. It also disallowed the addition of a new customer with an email that is associated with an already existing customer. However, we recognised a problem - there was no way to add credit for a customer whose record exists in the database but currently has zero credit (and thus is not displayed in this section), although he's not actually a 'new' customer.

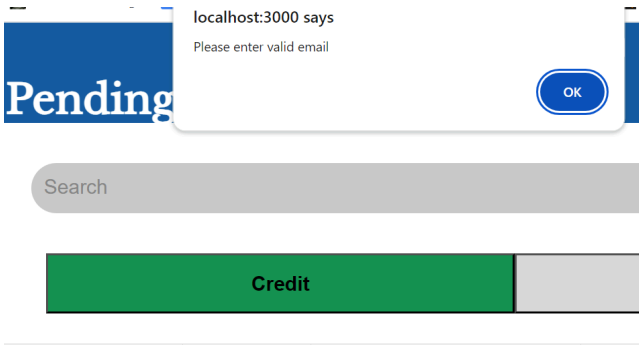
Debugging: We resolved the bug by using a prompt that pops up whenever the user tries to add a new customer with an existing email, asking if he would like to apply this change to the already existing record.

Structural Coverage: Functional coverage (covers the AddNewEntry() and ExistingEntry() functions in the frontend), Branch Coverage (various possibilities based on validity of phone number, validity of email, validity of amount and existing email).

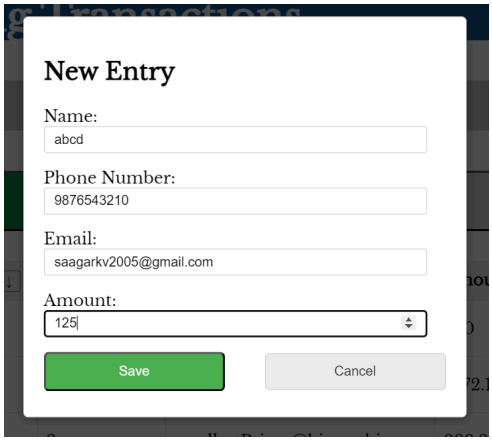
Within this, **multiple test cases** were used to test various possibilities. Some of them are shown here.

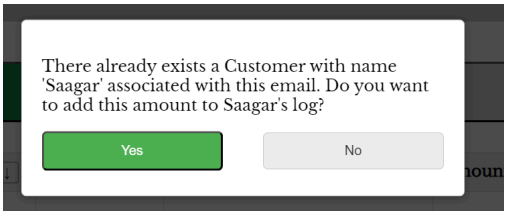
i) Invalid email:



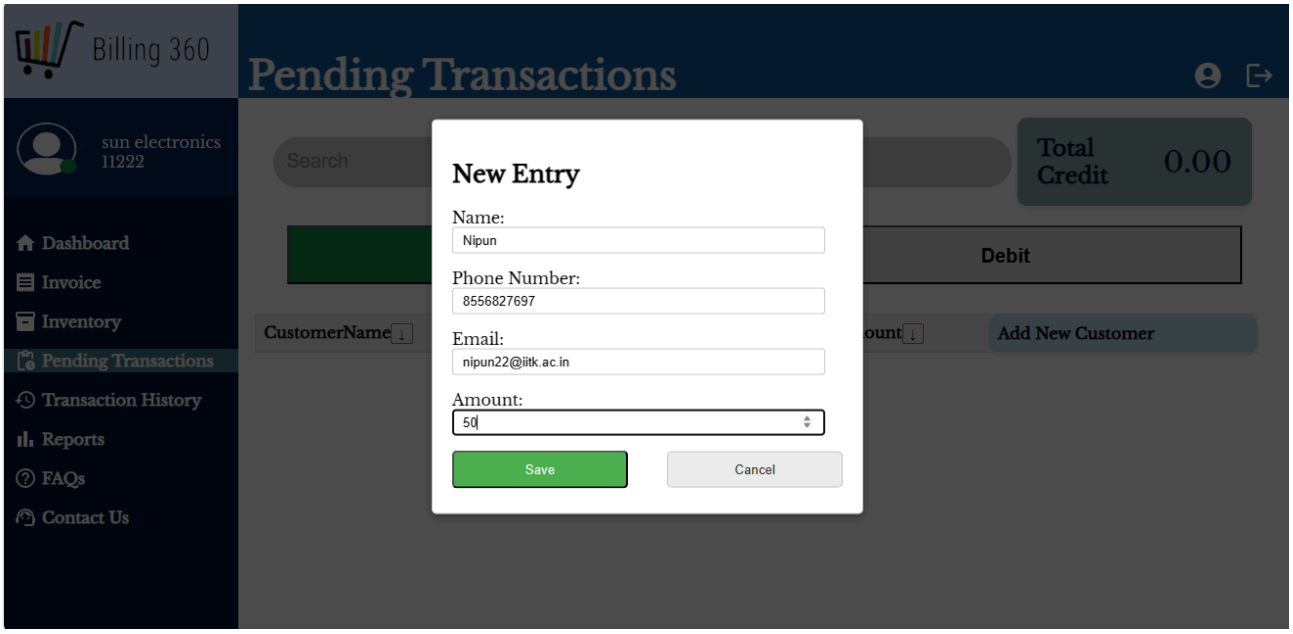


ii) Existing Email:





iii) Valid details entered:



JSON Object Logged onto the console:

d) Update Customer

In this unit, we have tested the updation of an existing customer in the database (backend testing). Frontend testing was also performed and results were as expected. It checked if the phone number and email are of a valid format and displayed alerts just like in the above case.

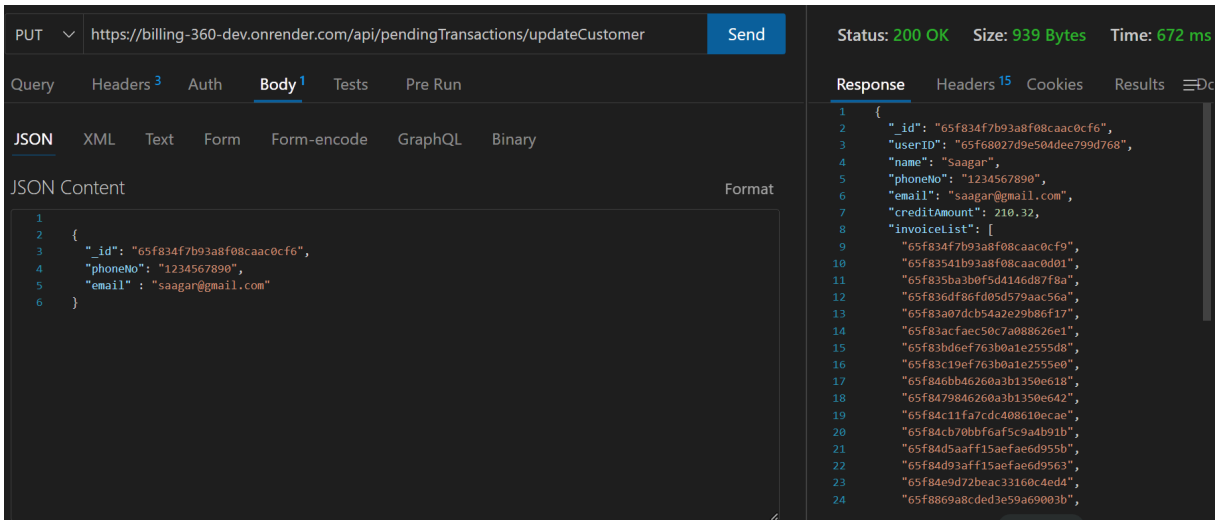
Unit Details: [Customer class, UpdateCustomer() function].

Test Owner: Saagar K V

Test Date: 28/03/2024

Test Results: The customer was correctly updated.

Structural Coverage: Functional coverage (covers the UpdateEntry() function in the frontend and the corresponding controller), Branch Coverage (various possibilities based on validity of phone number and validity of email).



e) Clear Dues/ Add Amount

In this unit, we have tested the updation of the pending credit amount of an existing customer in the database (backend testing). Frontend testing was also performed and expected results were obtained. It was checked if the amount was a valid positive number. An alert was displayed otherwise.

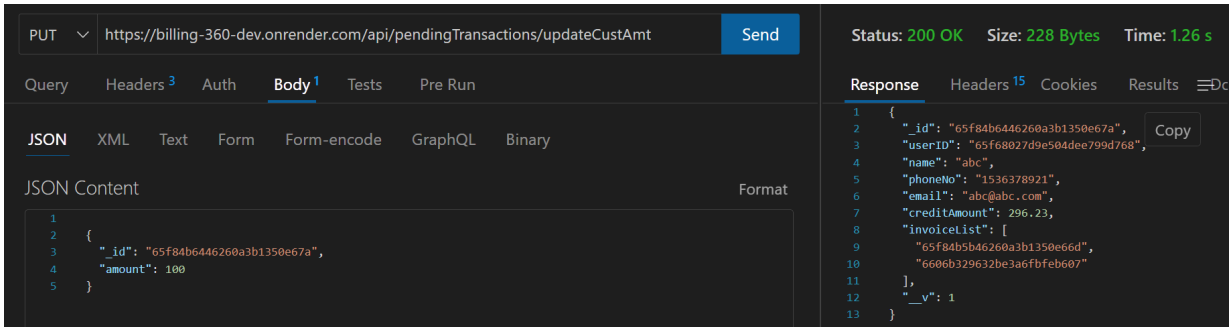
Unit Details: [Customer class, updateAmount() function].

Test Owner: Saagar K V

Test Date: 28/03/2024

Test Results: The customer amount was correctly updated.

Structural Coverage: Functional coverage (covers the updateAmount() function)



Note:

- The Supplier class also uses the same functions for above operations (polymorphism). The function chooses between customer and supplier based on the value of a variable ‘entryType’. Thus, testing of one case automatically completes the testing of the other. However, both cases were tested separately and similar results were obtained.
- Features such as the sending of email and adding invoice to customer list were tested in the ‘Invoice’ section. However, it was verified that those functions are correctly invoked, from this section, and correct results were obtained.

f)Search by Name

In this unit, we have tested whether the search bar filters the customers correctly.

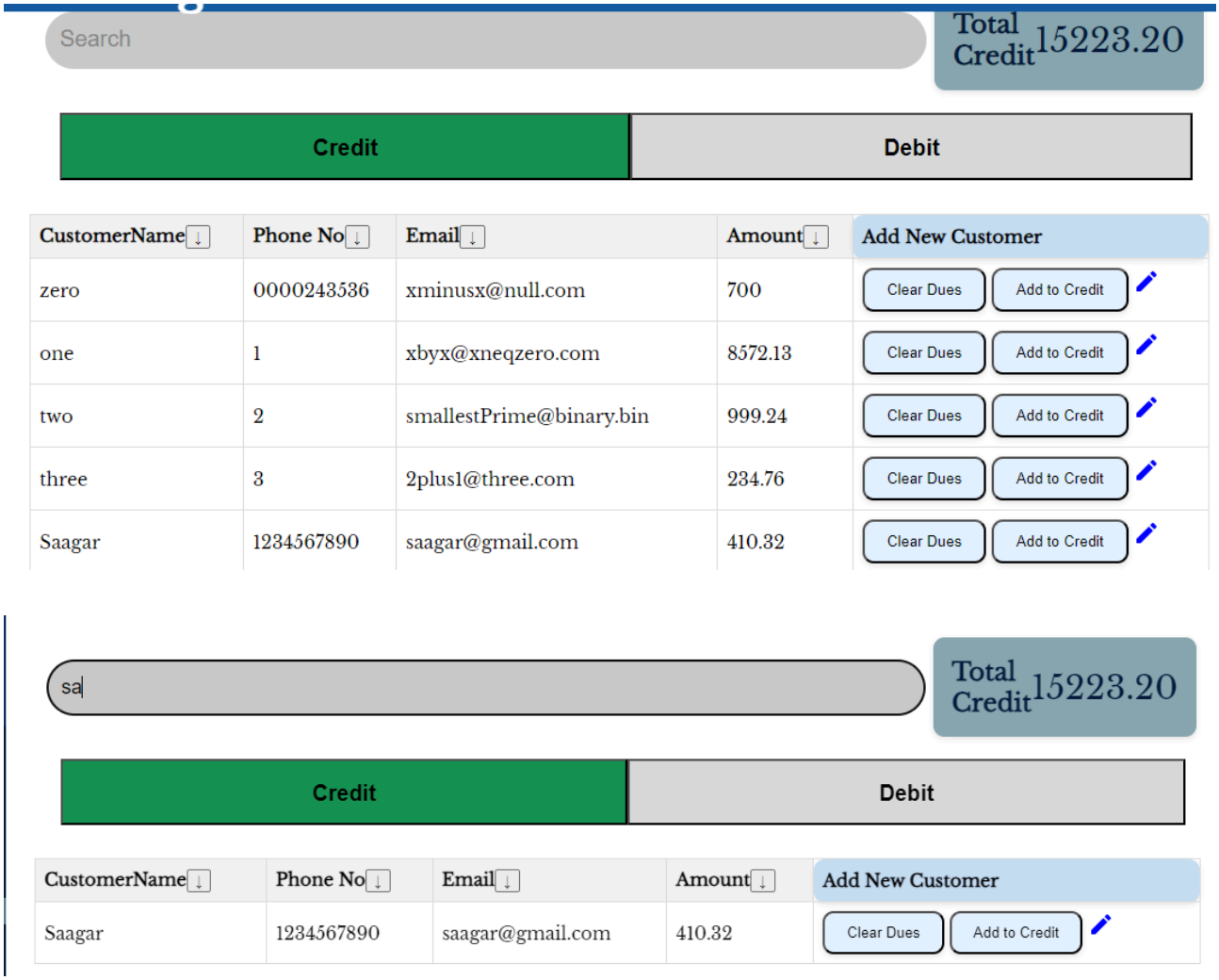
Unit Details: [Customer class, SearchCreditCustomers].

Test Owner: Nipun

Test Date: 28/03/2024

Test Results: The customers corresponding to the search parameters are shown..

Structural Coverage: Functional coverage (covers the SearchCreditCustomers() function)



6. **Transaction History**

Search by Customer Name

In this unit, we have tested the search feature. Upon searching a customer name, all transactions associated with that customer show up, hiding the others. Note that we have

used search using regular expressions, thus all customers whose names contain the given pattern (case insensitive) will be considered for display (backend testing).

Unit Details: [Invoice class, searchInvoice() function]

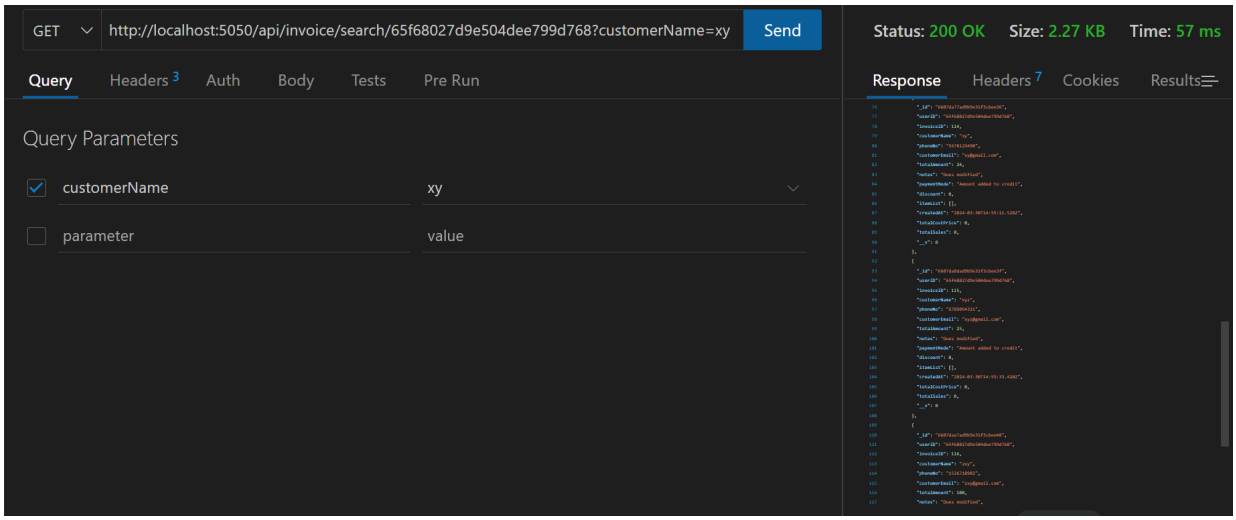
Test Owner: Saagar K V

Test Date: 28/03/2024

Test Results: The search was successful and worked as expected.

Structural Coverage: Functional coverage (covers the searchInvoice() controller function)

Frontend testing was also performed. The search query entered by the user was correctly taken and sent in the request to the backend. The data displayed to the user matched with the response obtained during backend testing, thus confirming the proper working of fetchSearchData() function in the frontend as well.



7. Inventory

a) Add new batch:

In this unit, we have tested whether a new batch is getting correctly added or not (backend testing).

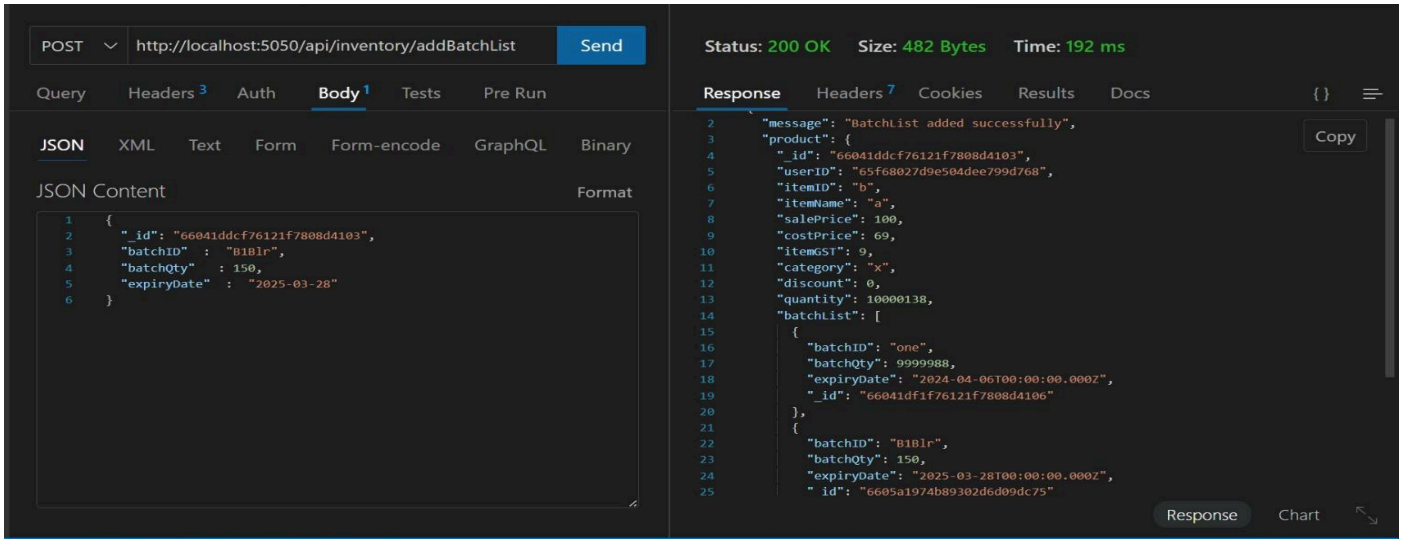
Unit Details: [Item class, addBatchList() function]

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: New batch is correctly added if all details are entered.

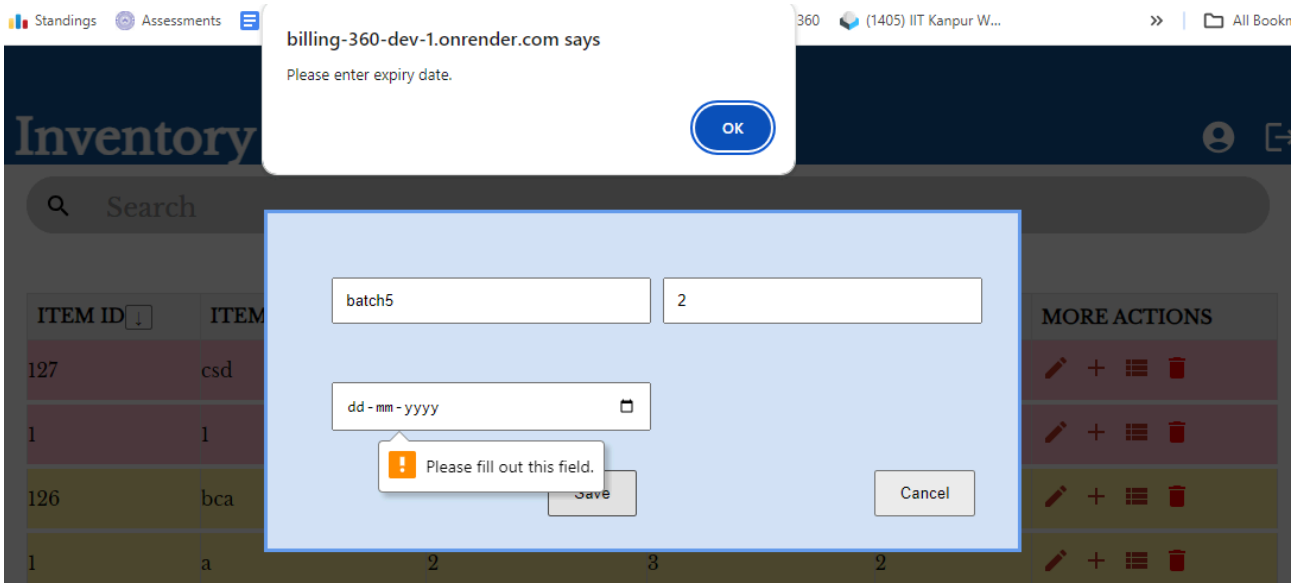
Structural Coverage: Functional coverage (covers the addBatchList() controller function).



Frontend testing:

I)If all details are not entered

Result:Message is displayed asking for missing details.

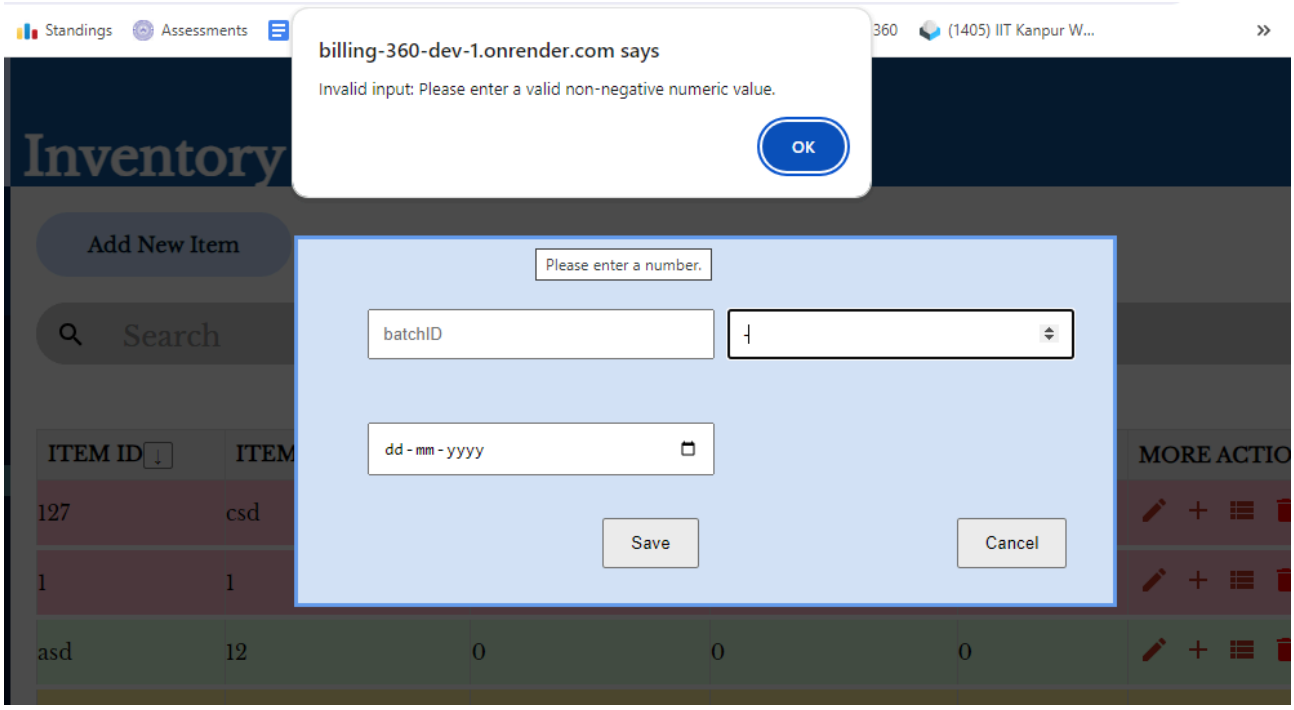


II)If invalid input is added (eg-alphabets for quantity)

Result:Invalid characters are not added in the input space.

III)If negative values are added for Number inputs

Result: Error message is displayed asking non-negative inputs.



b) Add new Product:

In this unit, we have tested whether a new product is getting added correctly or not (backend testing).

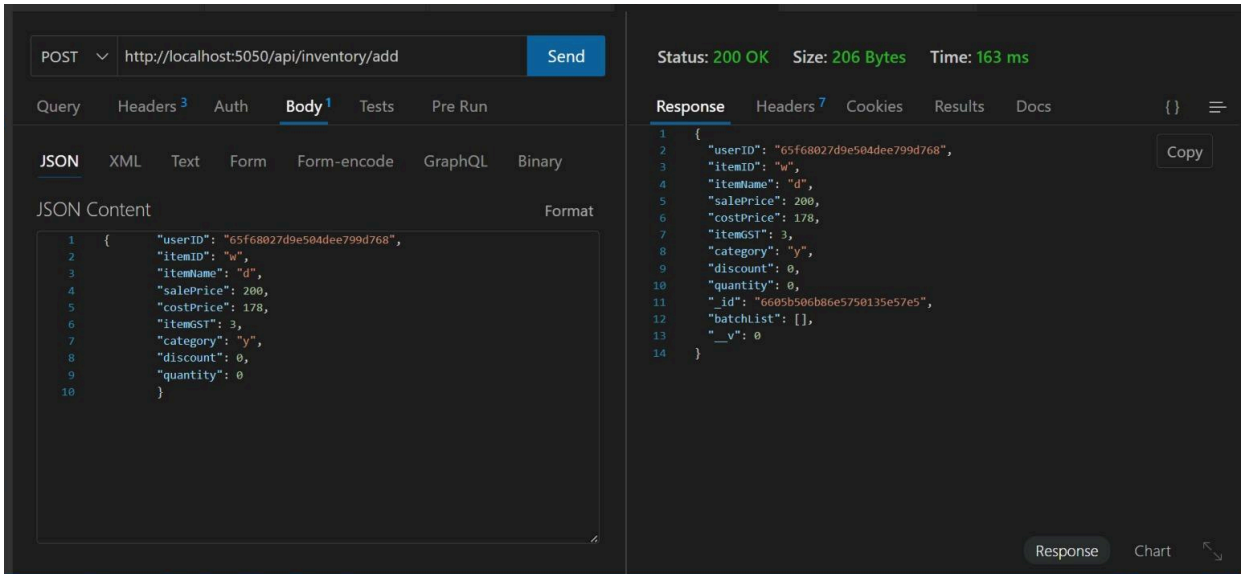
Unit Details: [Item class, addProduct() function].

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: New product is correctly added to the database.

Structural Coverage: Functional coverage (covers the addProduct() controller function).

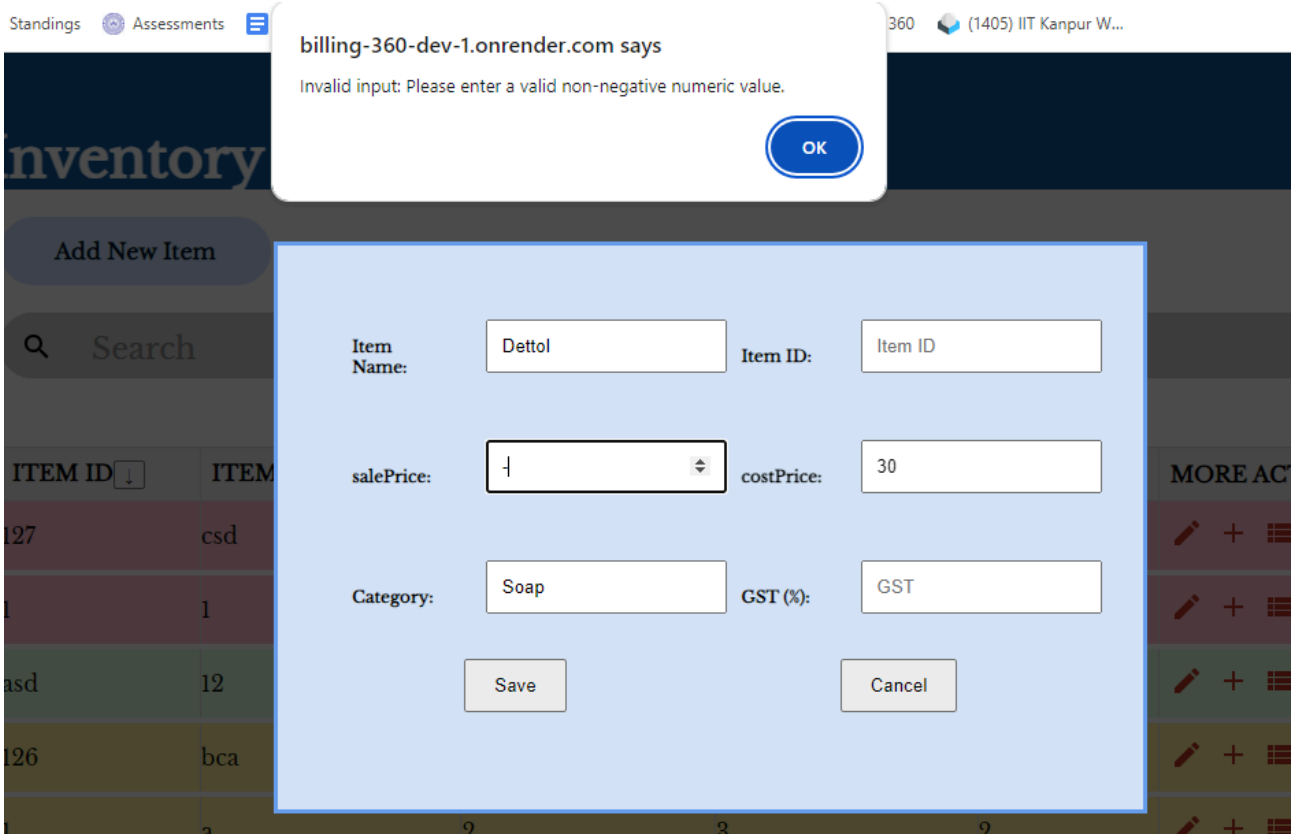


Frontend:

I)If invalid details are entered. (for example, alphabets for GST)

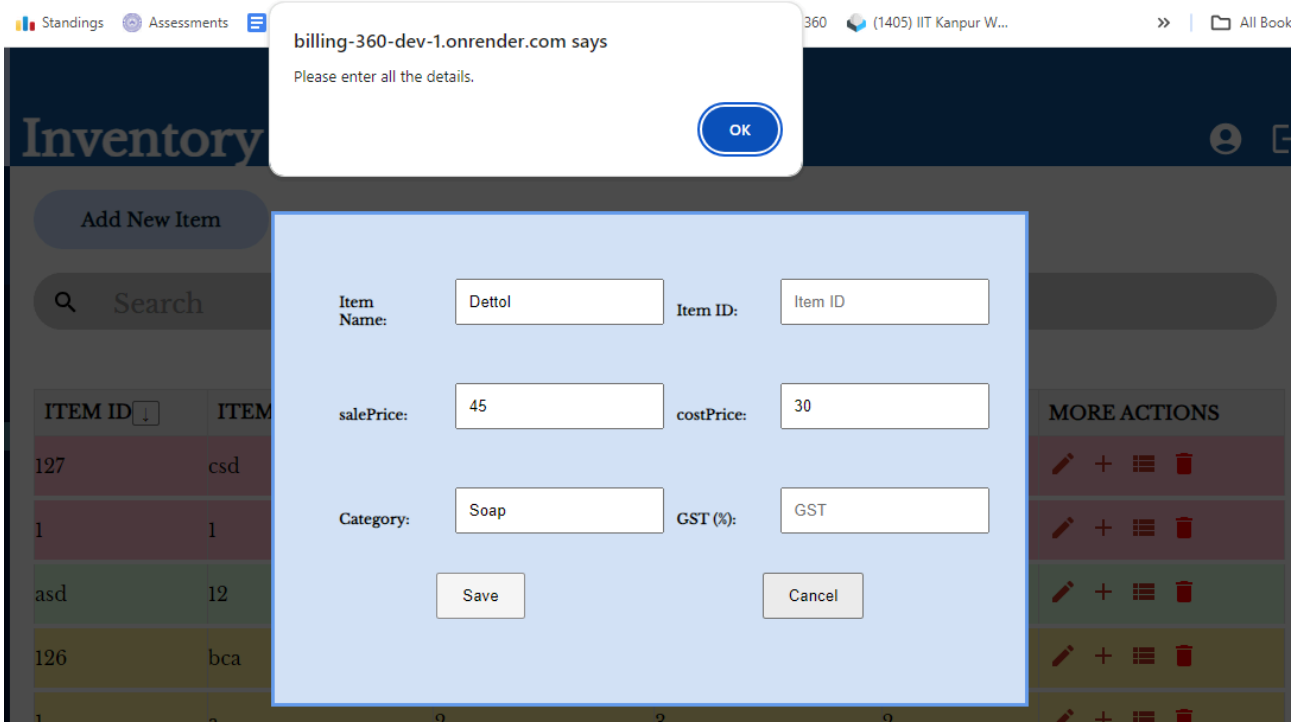
Result:Invalid characters are not added in the input space.

For Negative values message is generated asking to put non-negative value.



II)If not all details are entered.

Result:Message is displayed asking to enter all details



C) UpdateProduct:

In this unit, we have tested whether a product is getting updated correctly or not (backend testing).

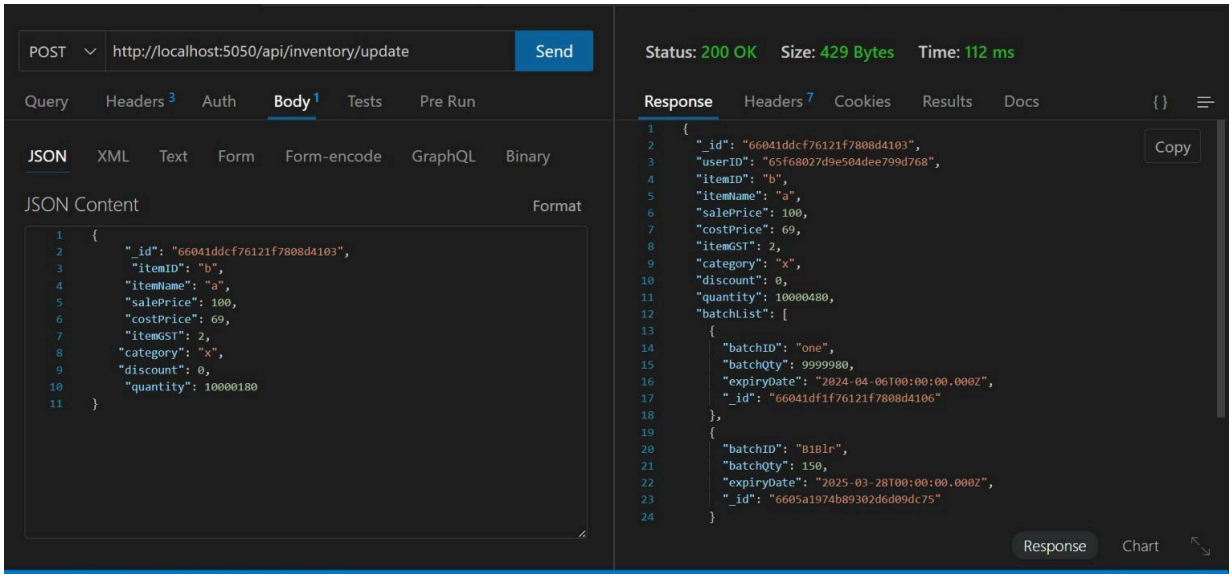
Unit Details: [Item class, updateProduct() function].

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: updateProduct() function correctly updates the data related to a given product.

Structural Coverage: Functional coverage (covers the updateProduct() controller function).



D) UpdateBatch:

In this unit, we have tested whether a batch is getting updated correctly or not (backend testing).

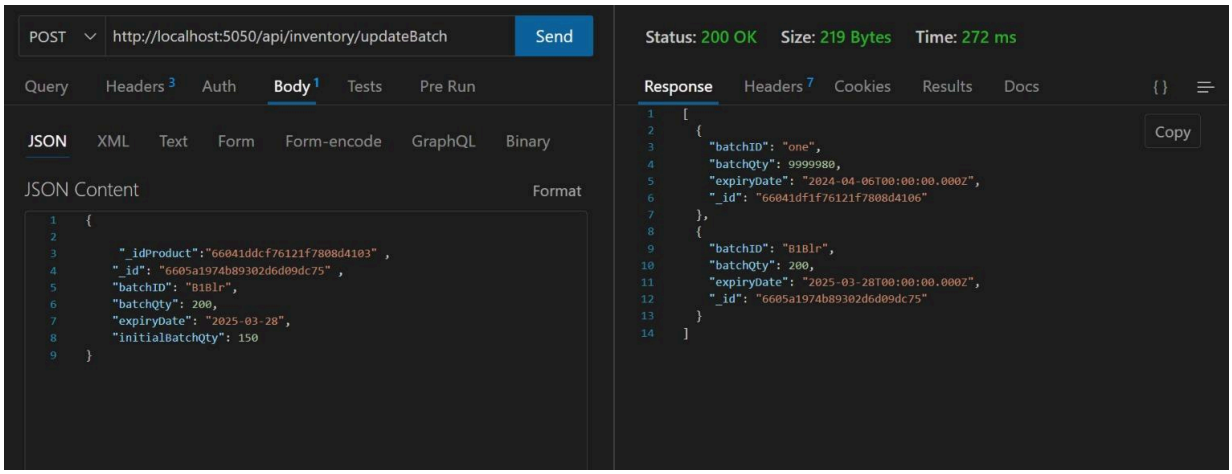
Unit Details: [Item class, UpdateBatch() function].

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: updateBatch() function correctly updates the data related to a given batch..

Structural Coverage: Functional coverage (covers the updateBatch() controller function).



D) View All Products

In this unit, we have tested whether the list of all products in a particular user's inventory is being extracted correctly from the database(backend testing).

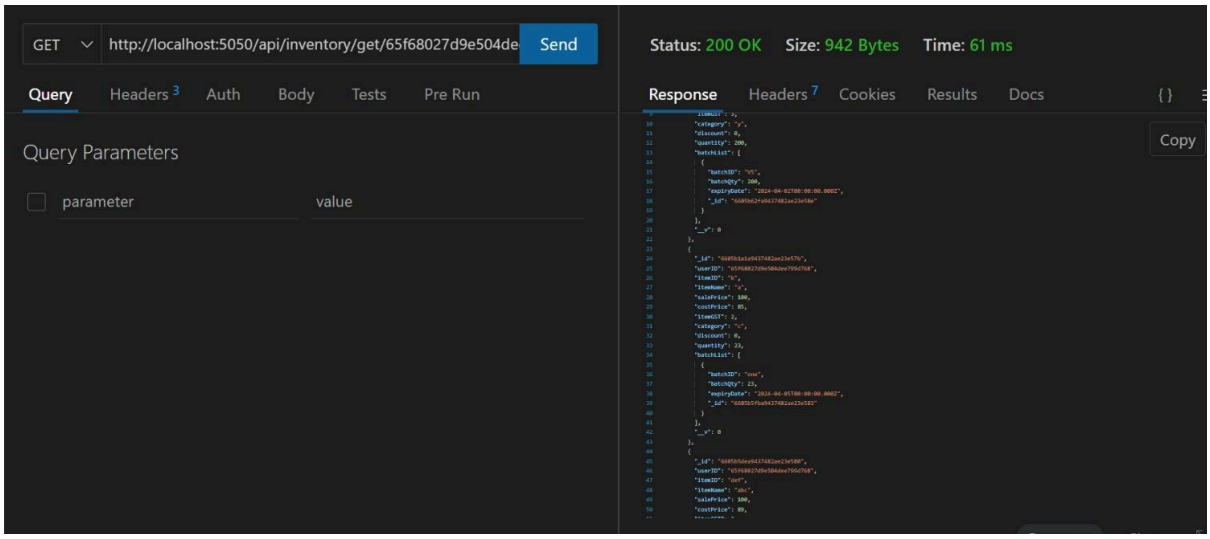
Unit Details: [Item class, getAllProducts() function]

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: All products in the user's inventory are correctly listed.

Structural Coverage: Functional coverage(It covers the getAllProducts() controller function).



E) Delete batch:

In this unit, we have tested whether a batch is getting deleted from the database correctly (backend testing).

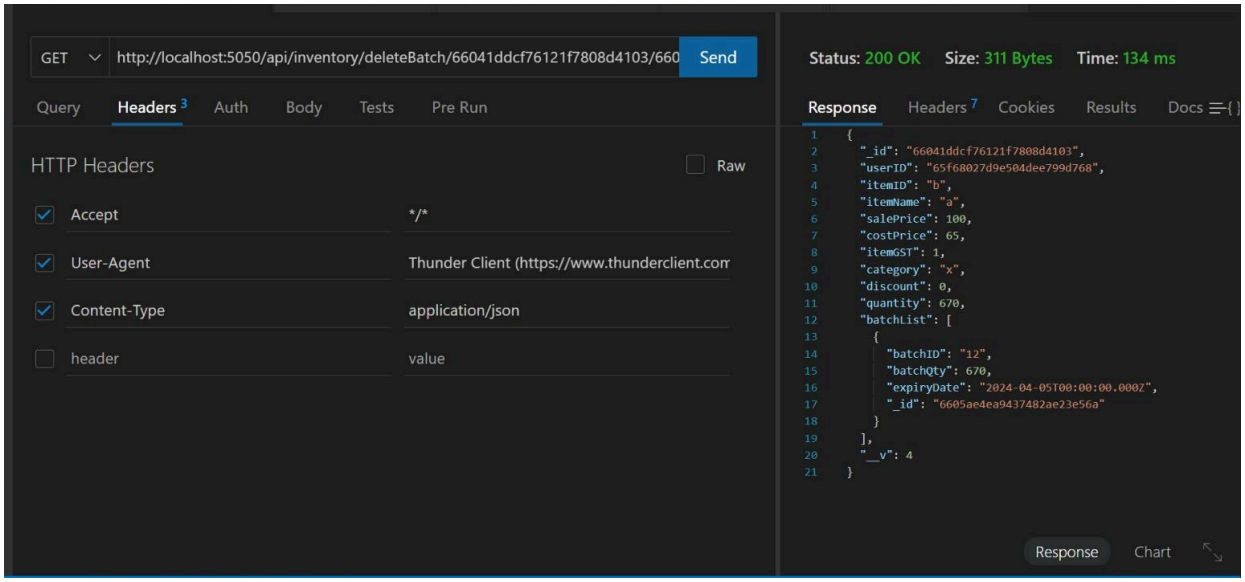
Unit Details: [Item class, deleteBatch() function].

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: The selected batch is correctly deleted.

Structural Coverage: Functional coverage (covers the deleteBatch() controller function).



F) Delete Product:

In this unit, we have tested whether the selected product is getting deleted from the database correctly (backend testing).

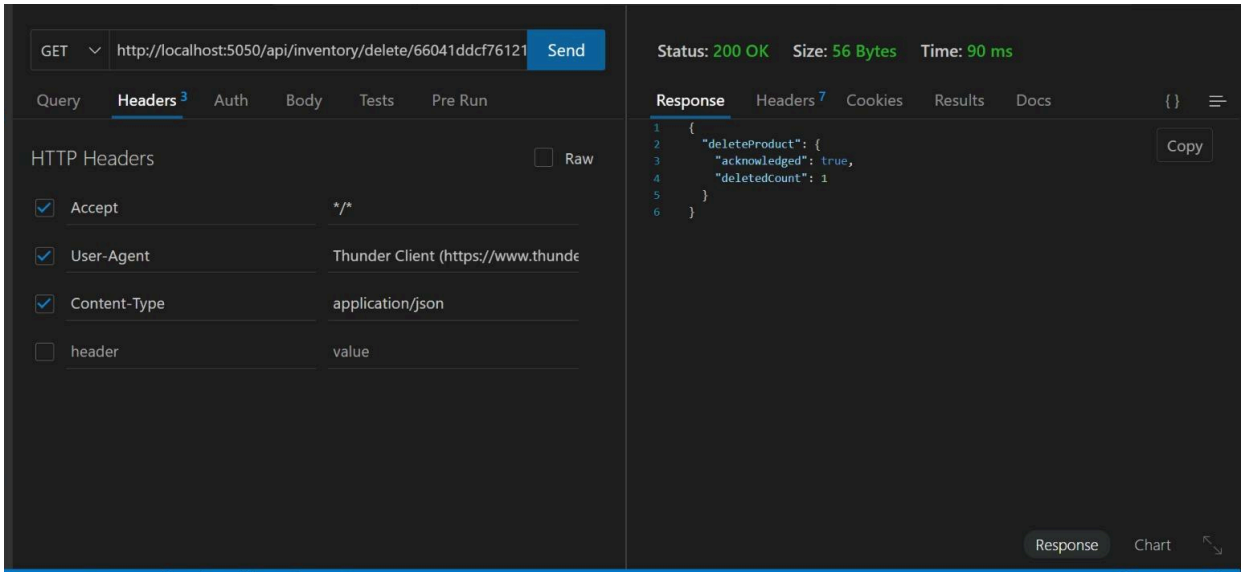
Unit Details: [Item class, deleteProduct() function].

Test Owner: Venkatesh

Test Date: 27-03-2024

Test Results: Selected Product is correctly deleted.

Structural Coverage: Functional coverage (covers the deleteProduct() function).



G) Sorting

In this unit, we have tested the sorting feature (frontend testing).

Unit Details: [Item class, sortProducts() function].

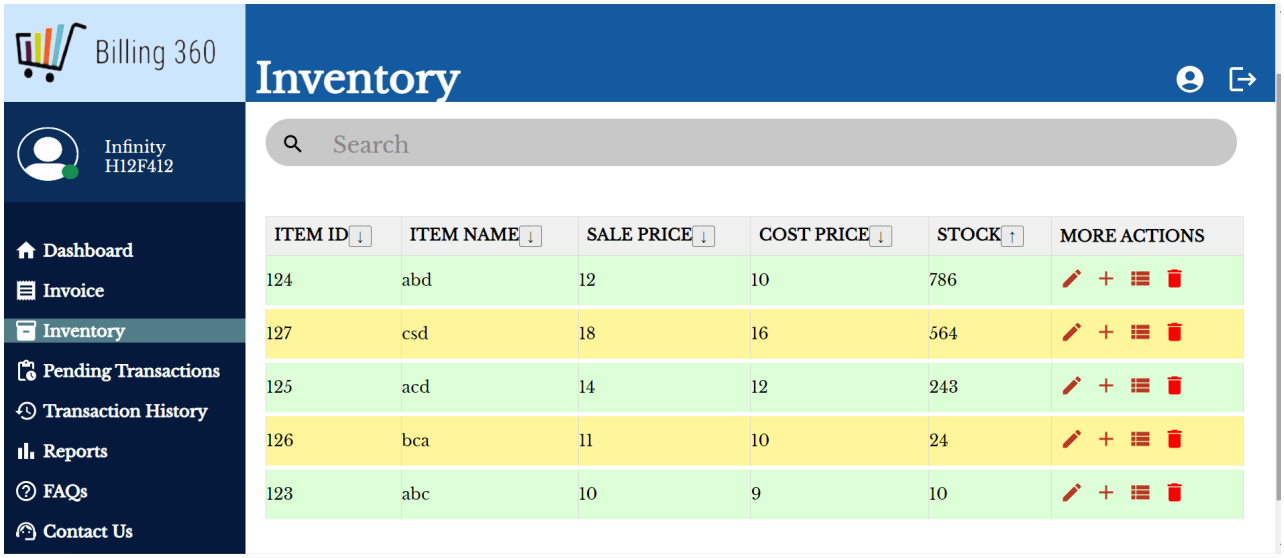
Test Owner: Nipun

Test Date: 28/03/2024

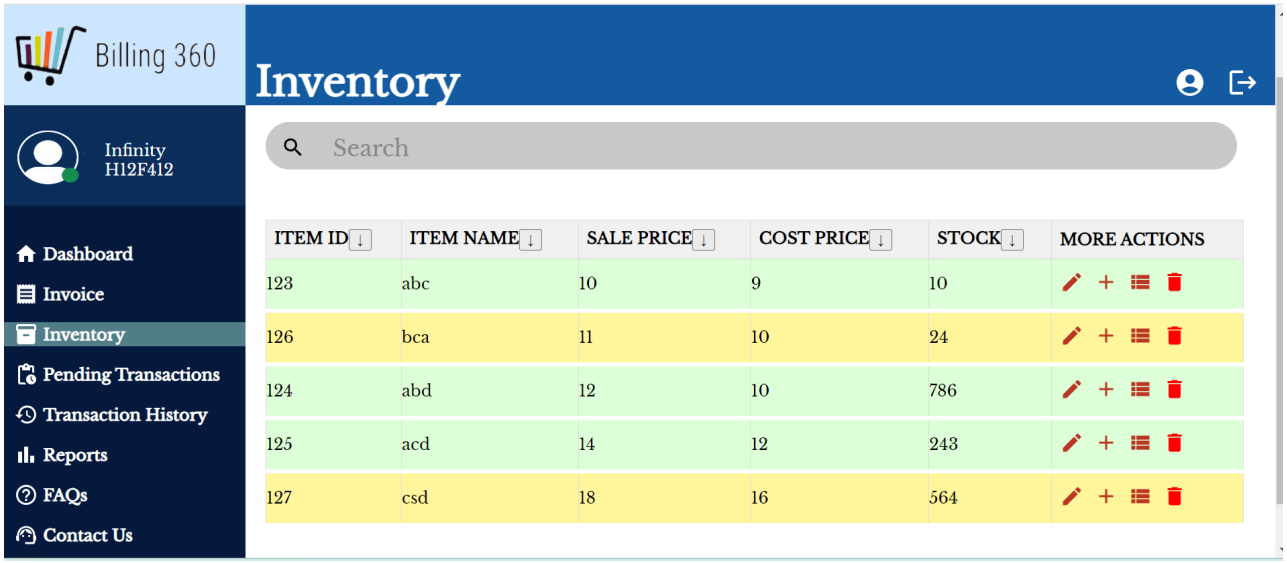
Test Results: Items were displayed correctly based on the chosen field and in the chosen order.

Structural Coverage: Functional coverage (covers the implementation of sorting using the sort() method along with states sortBy, sortDirection and sortedProducts in the Inventory component's sortProducts() function). Two examples are shown below:

i) Sorting in decreasing order of stock



ii) Sorting in increasing order of sale price



H) Colour coding of items based on expiry date

In this unit, we have tested whether the items are color coded according to expiry dates(red for expired, yellow for near expiry (expires after 3 days), green for fresh) (frontend testing).

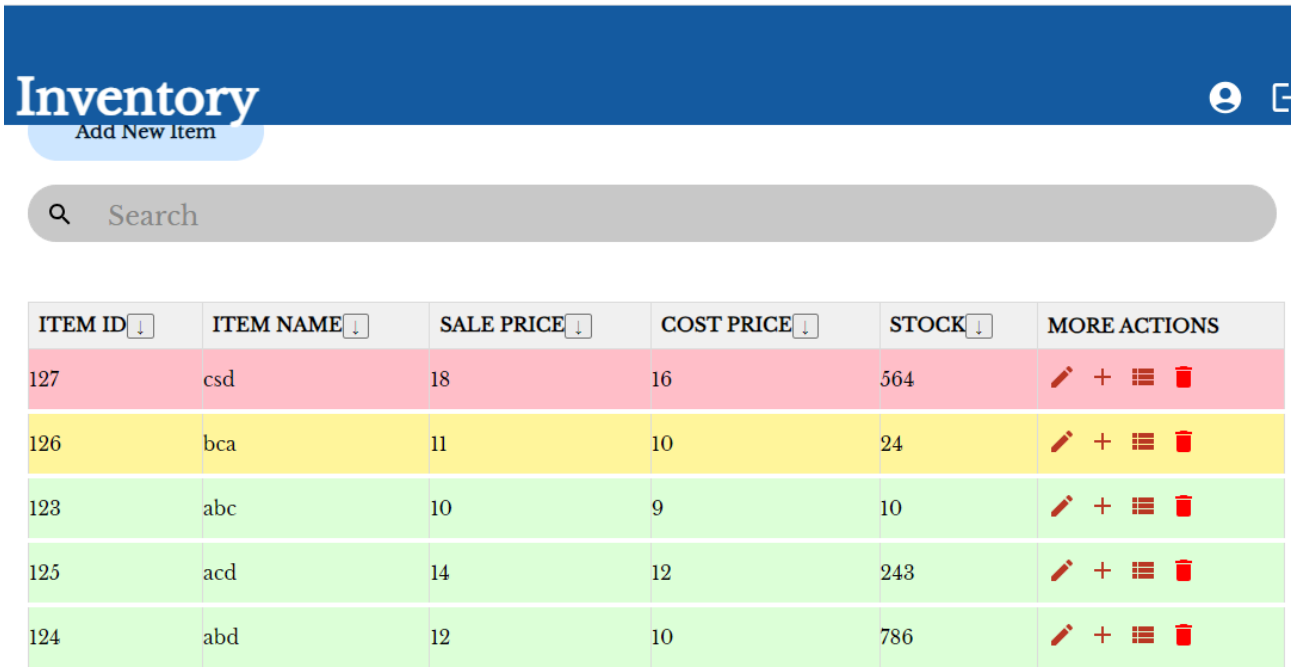
Unit Details: [Item class, getRowStyle() function]

Test Owner: Nipun

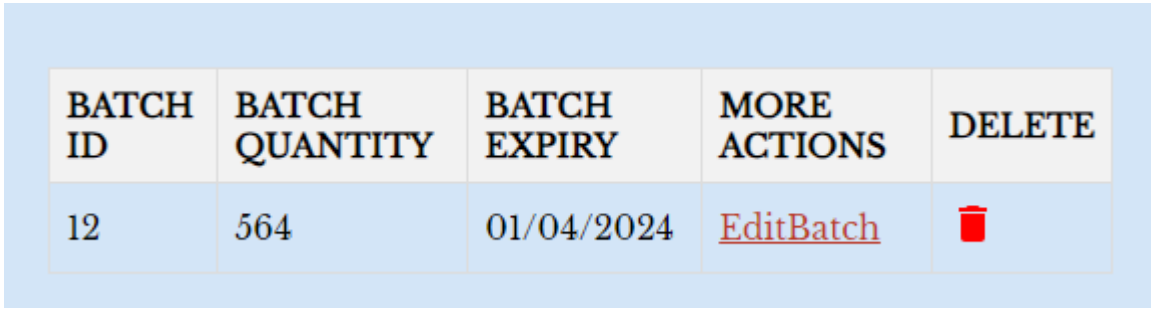
Test Date: 27-03-2024

Test Results: Products are correctly color coded according to the expiry date.

Structural Coverage: Functional coverage (covers the getRowStyle() function within the Inventory component in the frontend).



Batches of items displayed above:



BATCH ID	BATCH QUANTITY	BATCH EXPIRY	MORE ACTIONS	DELETE
12	24	02/04/2024	EditBatch	

BATCH ID	BATCH QUANTITY	BATCH EXPIRY	MORE ACTIONS	DELETE
1	10	06/04/2024	EditBatch	

BATCH ID	BATCH QUANTITY	BATCH EXPIRY	MORE ACTIONS	DELETE
29	243	25/04/2024	EditBatch	

BATCH ID	BATCH QUANTITY	BATCH EXPIRY	MORE ACTIONS	DELETE
123	786	26/04/2024	EditBatch	

l) **Search:**

In this unit, we have tested whether the search bar works correctly(frontend testing)

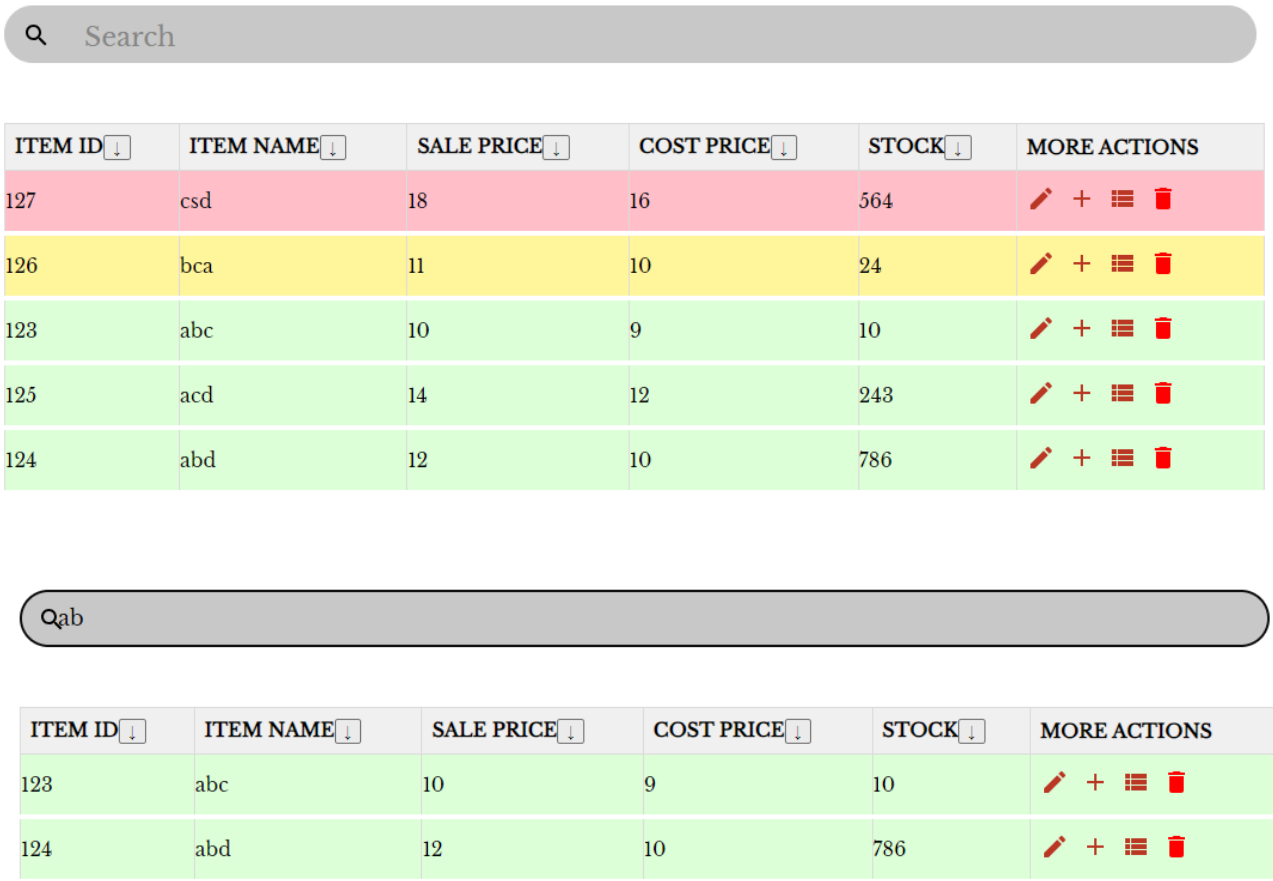
Unit Details: [searchInput, setSearchInput]

Test Owner: Nipun

Test Date: 27-03-2024

Test Results: Search bar outputs all items corresponding to search parameters.

Structural Coverage: Functional coverage (covers the setSearchInput() function within the Inventory component in the frontend).



8. Reports and Charts

(a) Backend testing:

In this unit, we have tested the fetching of invoices generated within a specified time period, which would be used by the frontend to generate reports.

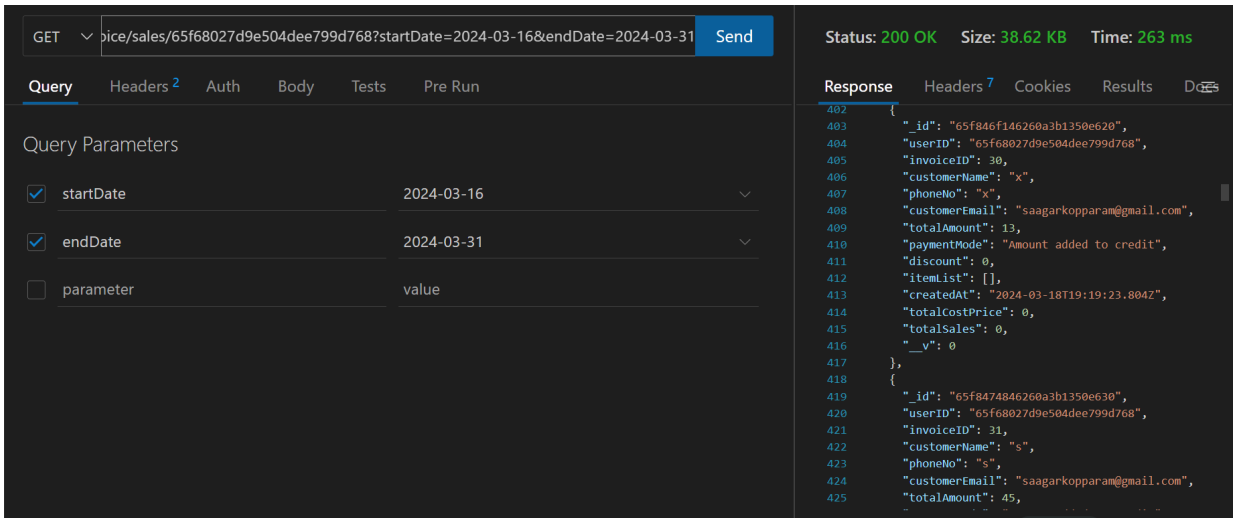
Unit Details: [Invoice class, getSalesData() function, getInvoiceCount, getAllInvouce, searchInvoice, getSalesData controller functions].

Test Owner: Nipun

Test Date: 28/03/2024

Test Results: Invoices were extracted from the database correctly.

Structural Coverage: Functional coverage (covers the getSalesData(), getInvoiceCount, getAllInvouce, searchInvoice, getSalesData controller functions).



(b) **Frontend testing:**

In this unit, we have tested the frontend of the reports section and the display of charts within it (frontend testing).

Unit Details: [Reports class, Reports() component]

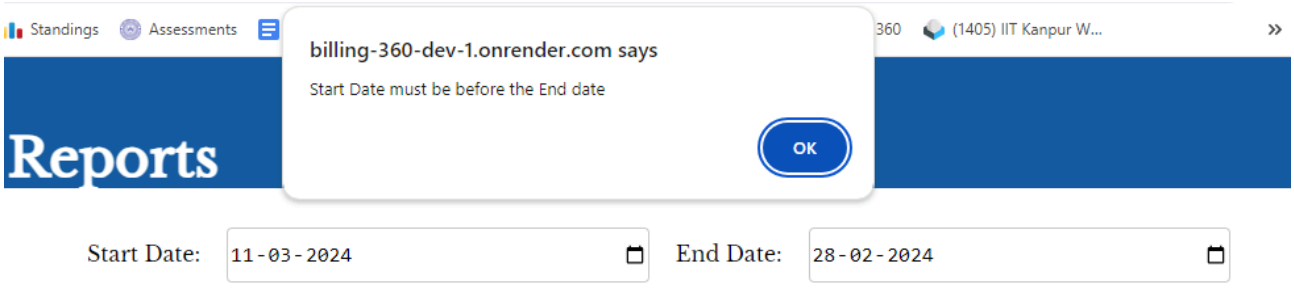
Test Owner: Nipun

Test Date: 27-03-2024

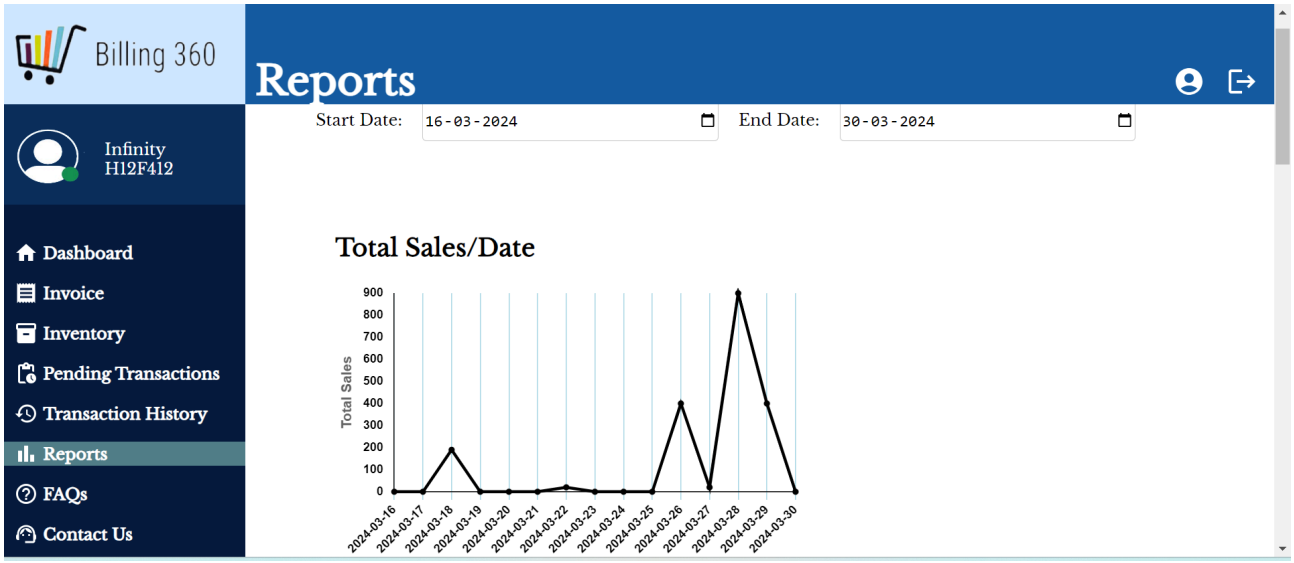
Test Results: The start date and end date were input as expected along with the check that confirms start date <= end date. Information like total sales, total profit etc was calculated correctly from invoice data and charts were displayed correctly using this information.

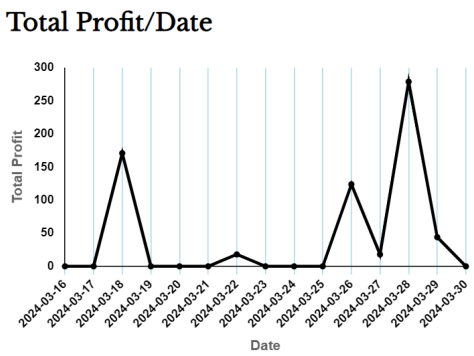
Structural Coverage: Covers the Reports() component and various components corresponding to different types of charts, present in the Charts folder.

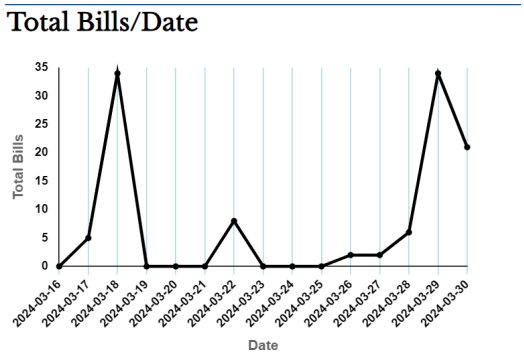
i)When start date is after end date



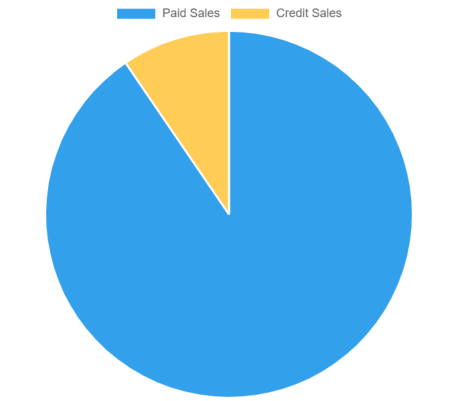
ii) When start and end date are entered correctly.







Sales Distribution



Top 5 Customers by Sales

Name	Email	Total Sales
Saagar	saagarkv2005@gmail.com	800
Saagar	saagarkopparam@gmail.com	300
xyz	a	200
qq	q	200
Venkatesh	akulav@gmail.com	200

Additional Comments: The graphs displayed in the dashboard are also similar. The code generating those graphs was also tested and was found to be correct.

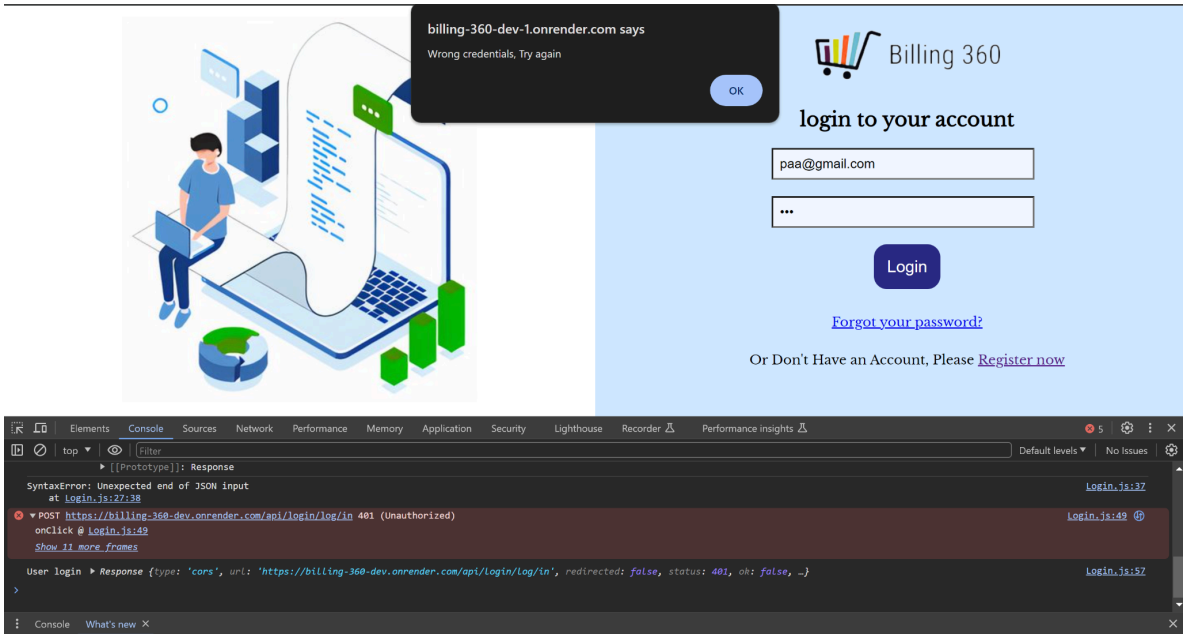
3. Integration Testing

Integration testing involves testing whether the backend and frontend interact the right way and produce the desired behaviour. Both backend and frontend units have been individually tested by the unit testing team. We have performed integration testing as follows. We gave the input to the frontend and observed final changes that occur after the backend processes the request sent by the frontend. Correct final behaviour implies that the interaction between backend and frontend was alright. Apart from this, integration testing also involves the interaction between different components, redirection of pages and updation of pages whenever a change is made to the database from anywhere.

1. Authentication

Module Details: The module encompasses testing procedures for user registration through OTP verification, login for already registered users, and the forgot password feature. Additionally, it includes validating the redirection to respective pages upon completion of these actions.
Test Owner: Poojal Katiyar
Test Date: 25/03/2024
Test Results: In this test we validated the successful integration of backend and frontend components of SignUp, Login, Email Verification and Forgot password feature. We also validated that database is updated on registration of new user and password is stored in encrypted format to maintain the security of the user.Here we are integrating login api, Sign Up api, Forgot Password api and generate otp api using frontend. Throughout the testing process, we also scrutinized the redirection of users to the correct pages upon the completion of these actions.

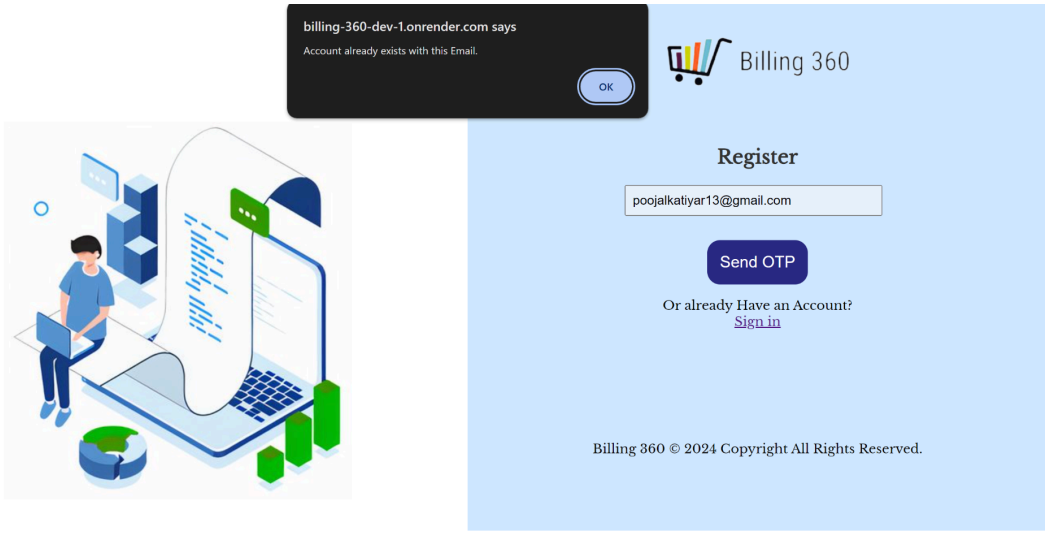
TEST #1:
Details: Invalid credentials entered while trying to login.



Result: Pop-up displaying that entered credentials are invalid. Login is denied.

TEST #2:

Details: Registering for an email which is already registered on the site.

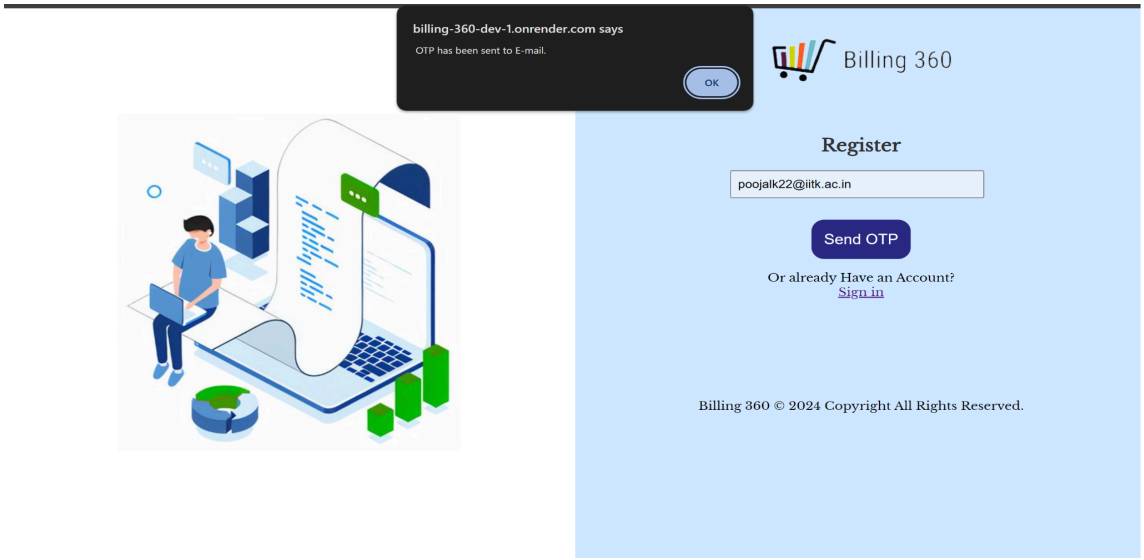


Result: Popup showing “Account already exists with this Email.”

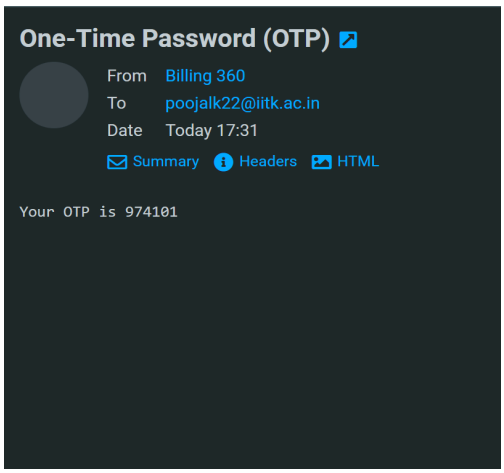
TEST #3:

Details:OTP Verification and registration of a new user.

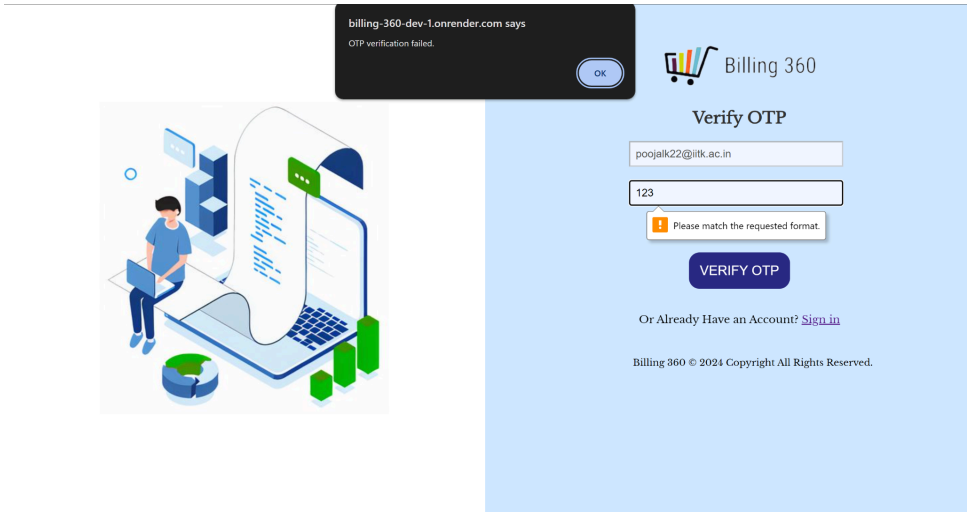
Flow: Email Entered in Register Page->OTP mailed->Directed to Verify OTP page->Directed to fill details Page where user registers himself with details->User Directed back to login page after registering successfully and user’s data updated in MongoDB database



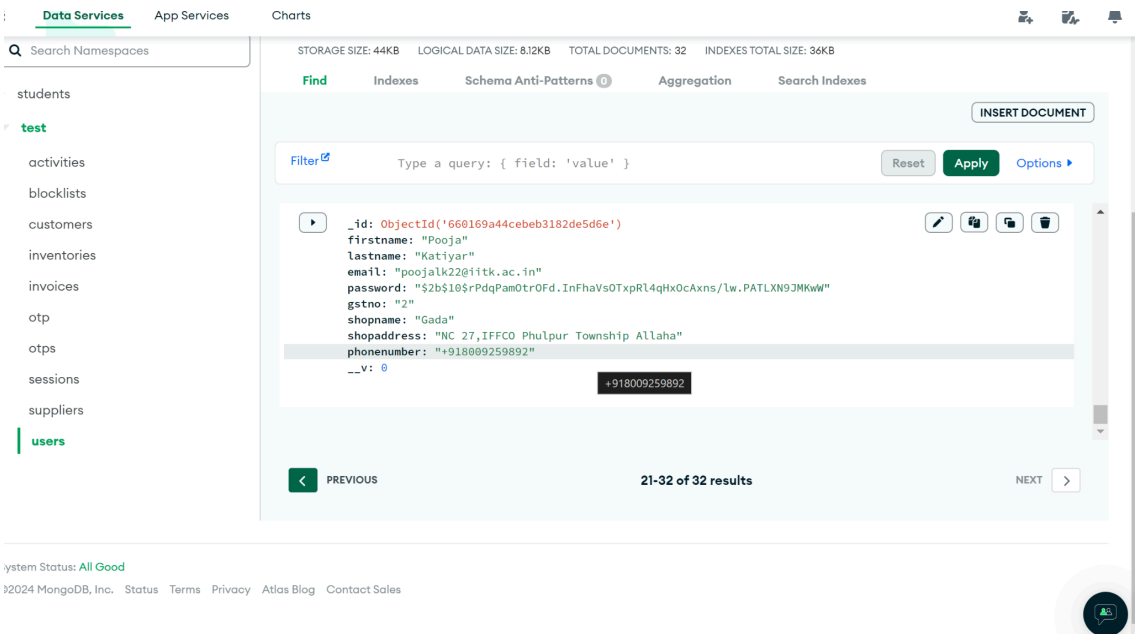
Mail sent for otp verification:



When otp entered does not matches with the sent otp:

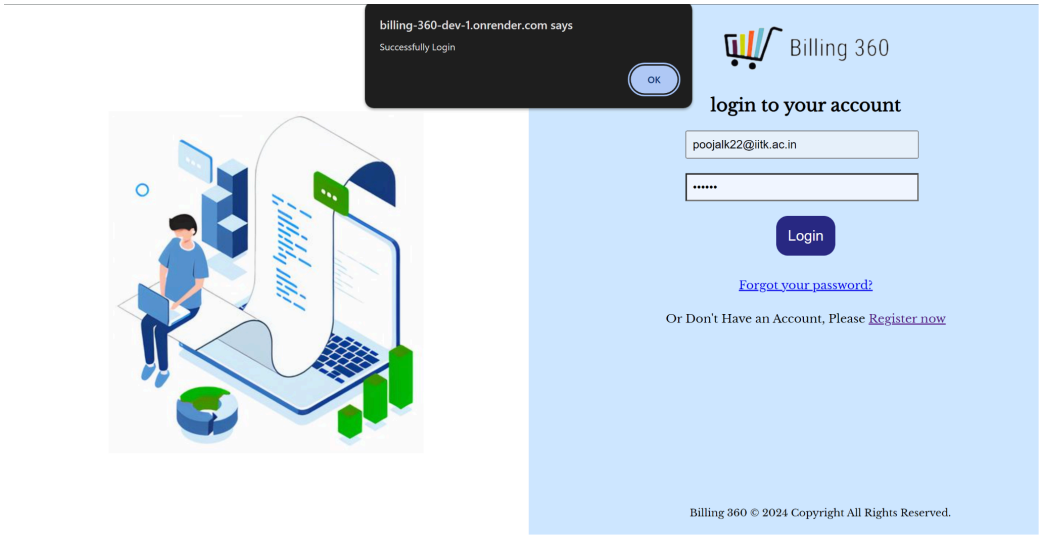


New User is updated in MongoDB Database, upon entering correct details and OTP.



Result:Finally when a new user enters everything correctly as described above he/she is successfully registered and ready to login.

TEST #5:
Details: Login with right credentials.

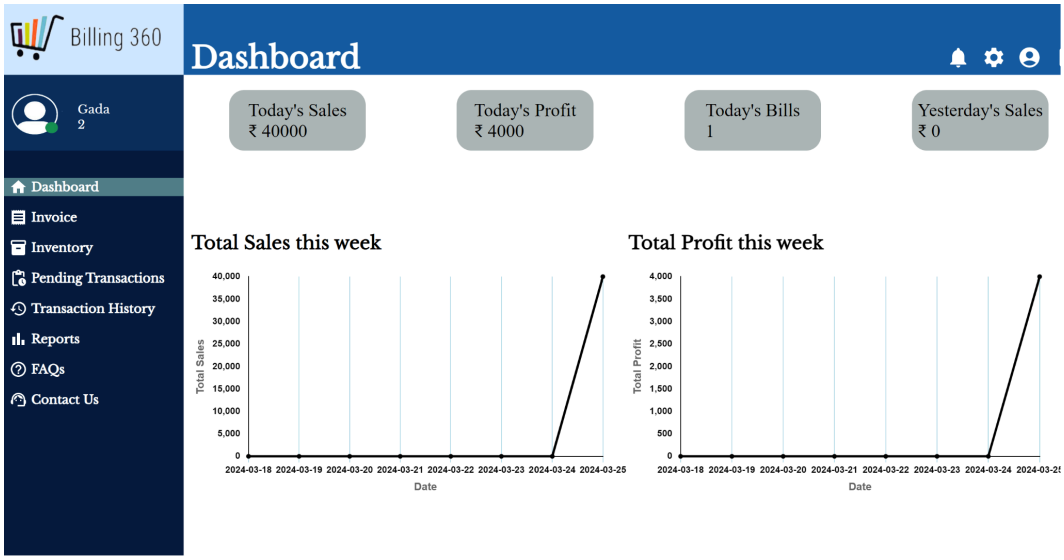


Frontend Console request for login :

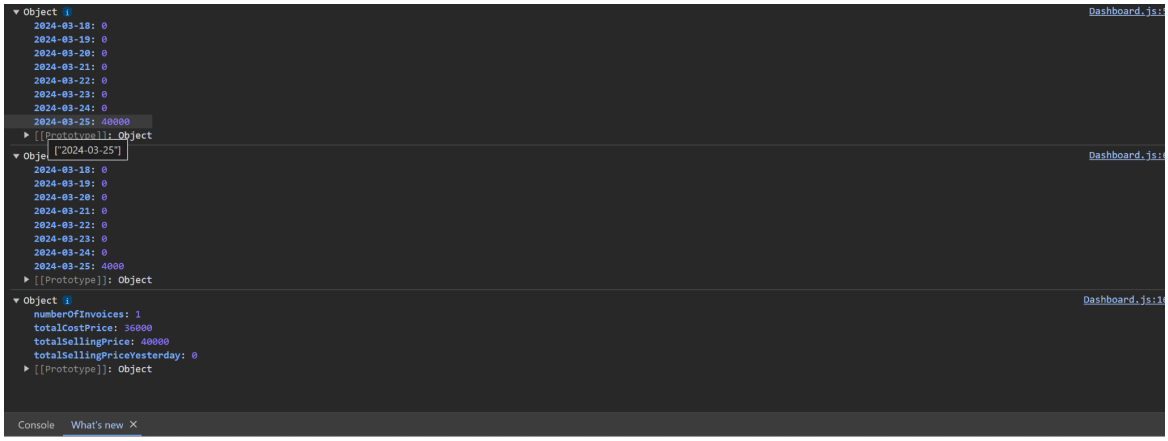
Backend Console which prints the user object :

```
{
  _id: new ObjectId('660169a44cebeb3182de5d6e'),
  firstname: 'Pooja',
  lastname: 'Katiyar',
  email: 'poojalk22@iitk.ac.in',
  password: '$2b$10$rPdqPamOtrOFd.InFhaVsOTxpRl4qHxOcAxns/lw.PATLXN9JMKwW',
  gstno: '2',
  shopname: 'Gada',
  shopaddress: 'NC 27,IFFCO Phulpur Township Allaha',
  phonenumber: '+918009259892',
  __v: 0
}
[]
today
```

After login user is directed to **Dashboard**

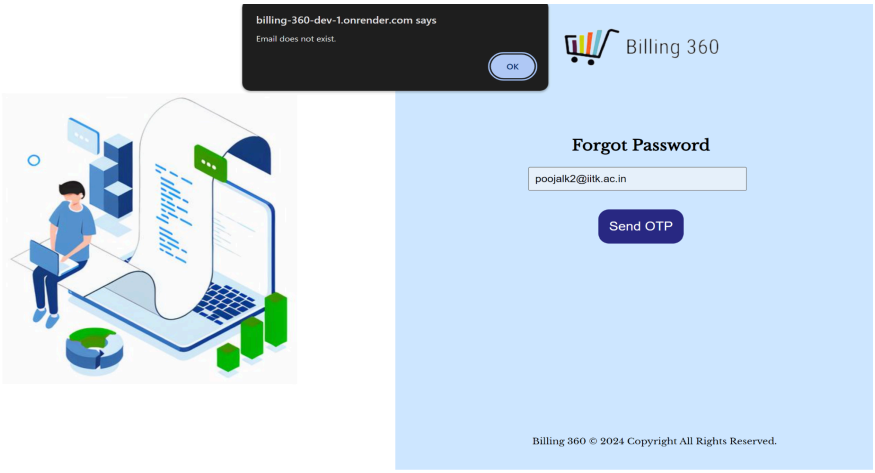


Frontend Console on logging into the site displays request to view on dashboard



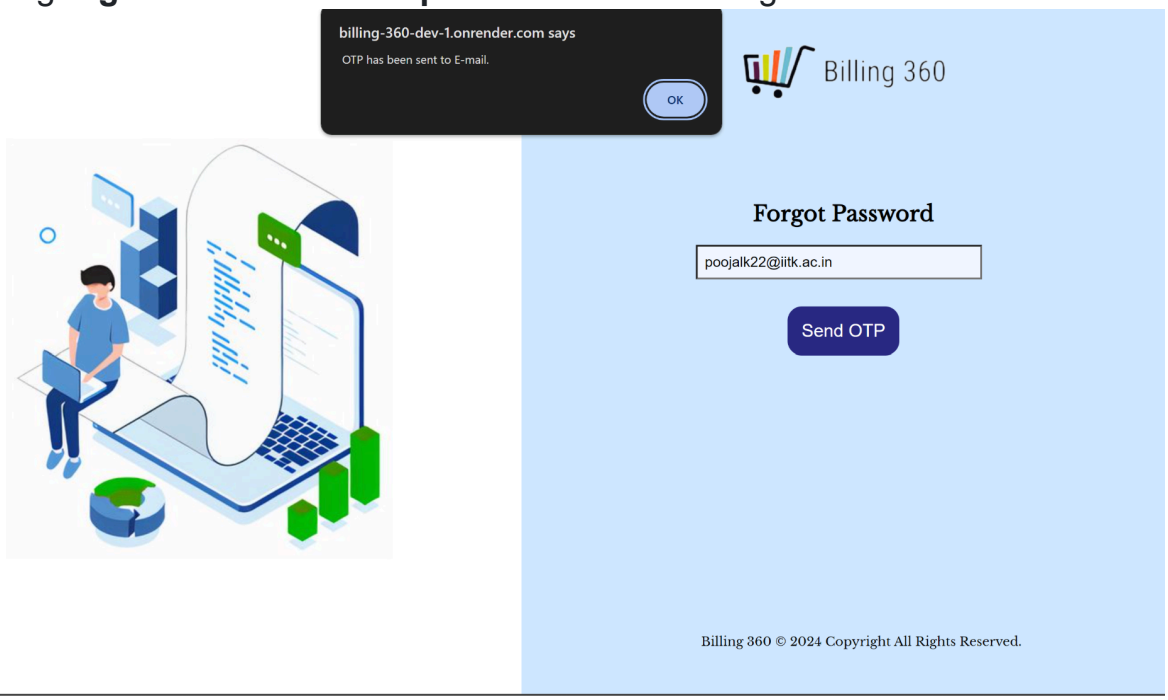
TEST #6:

Details: To verify the feature of forgot password and to check if the password is updated correctly.



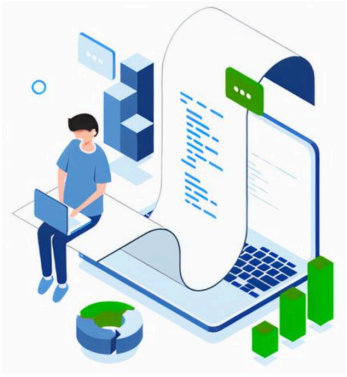
Result: Pop up box which displays “Email does not exist”

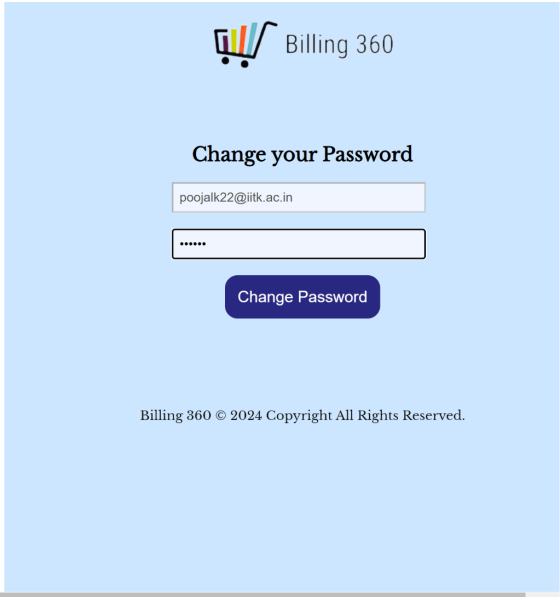
On entering **registered email-ID otp** will be sent to the registered email



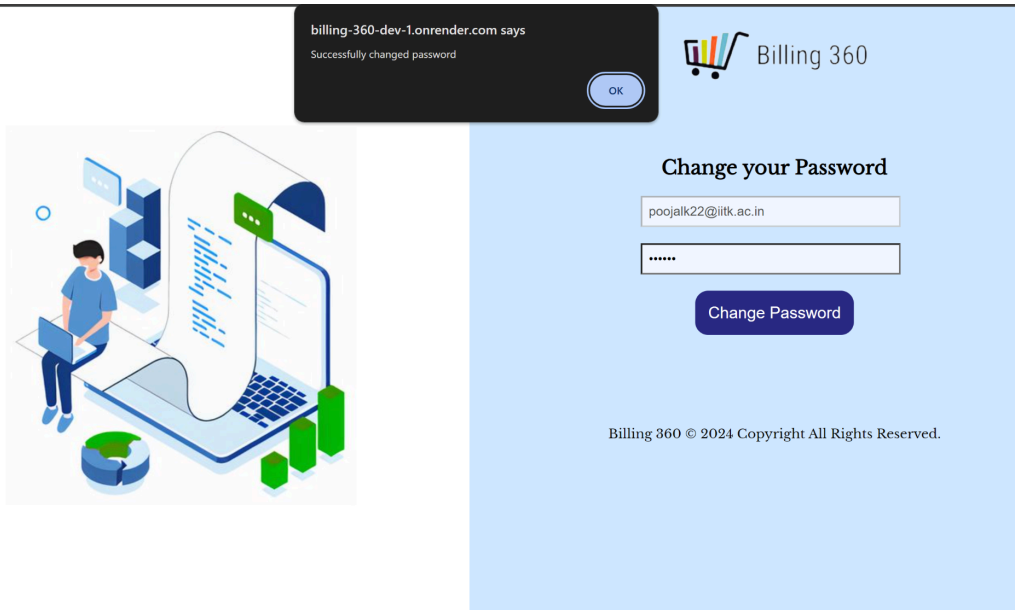
Result :OTP sent to the email will be entered and on clicking verify OTP will be directed to change your password page.

Change your password page:



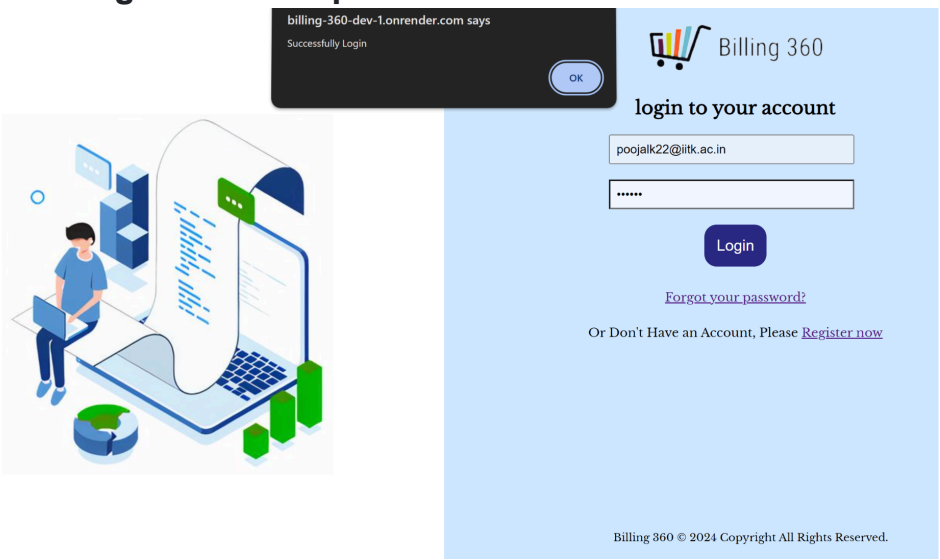


Result: After Entering new password pop up box which says password changed successfully.



Result: On clicking OK will be directed again to login page and now I can login with a new password.

Successful login with new password:



2. Inventory

Module Details: The module encompasses testing procedures for inventory management specific to a particular user which includes api calls for getting the data of all the products, adding a new item, updating and deleting an existing item in the inventory. Additionally, it also includes viewing all the batches of a particular item, adding a batch, editing and deleting an existing batch of an item.

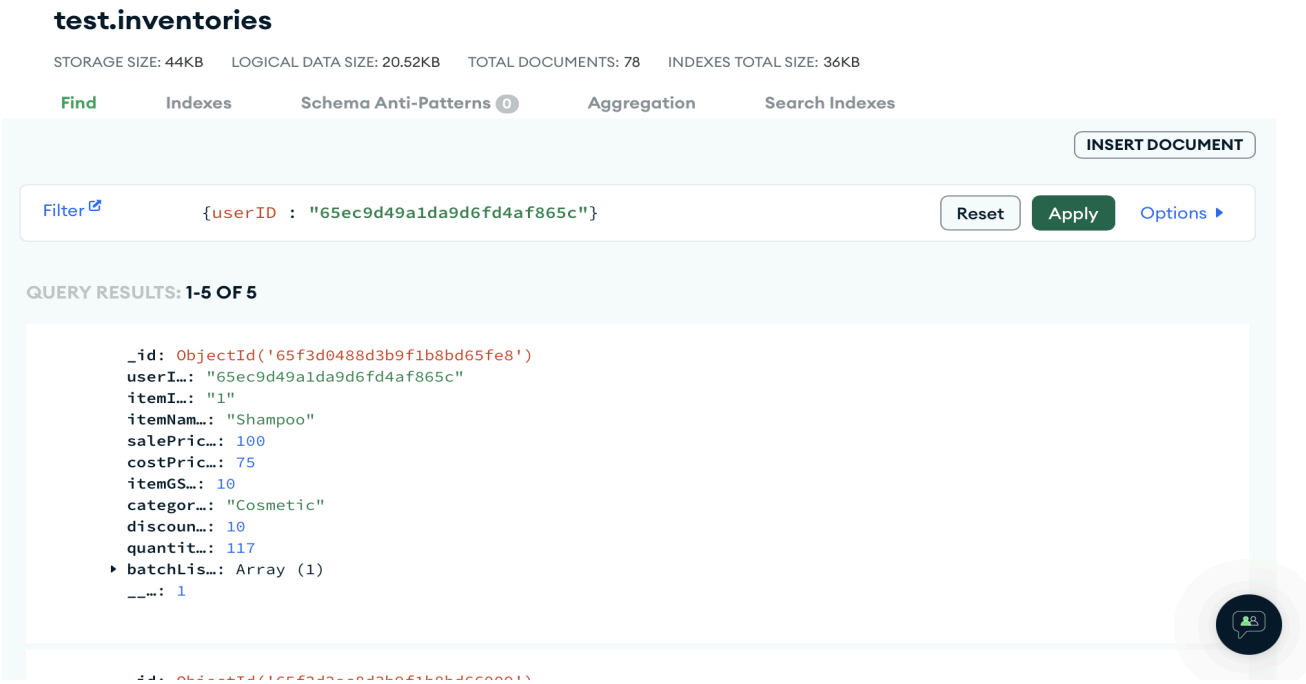
Test Owner : Abhishek Khandelwal
Test Date : 25/03/2024

Test Results : In this test we validated the successful integration of backend and frontend components of getAllProducts, addProduct, updateProduct, viewBatch, updateBatch and delete(Batch/item). Apart from this, we successfully tested that the stock field in the inventory page represents the correct total items over all the batches after all kinds of modifications. Throughout the testing process, we also scrutinized the redirection of users to the correct pages upon the completion of these actions.

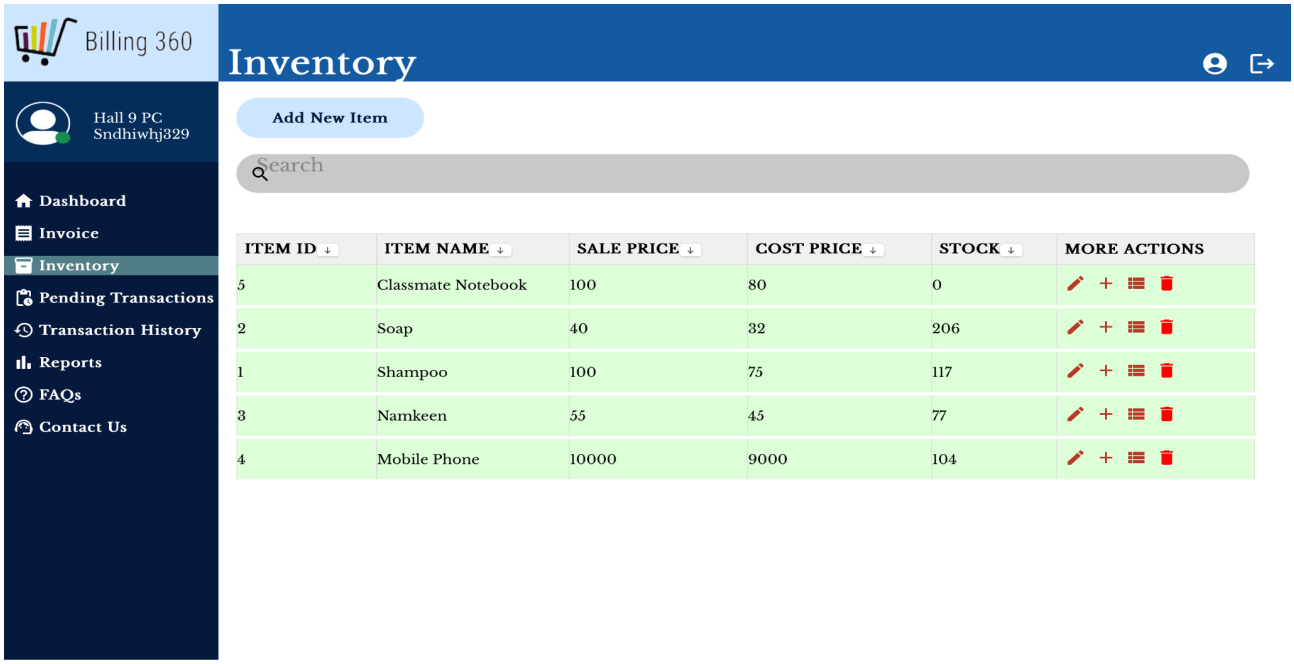
Test #1

Details : To verify whether all the products in the inventory of a particular user are fetched correctly or not.

Searching for all the products of the authorised user in the database for verification.



Verifying whether the displayed data is in correspondence with the database data or not.

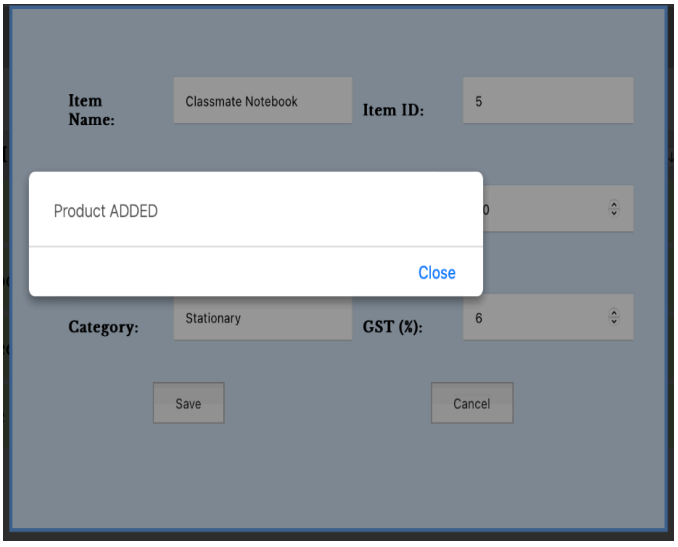


Result : Displaying all the products in the inventory with correct details of the user logged in.

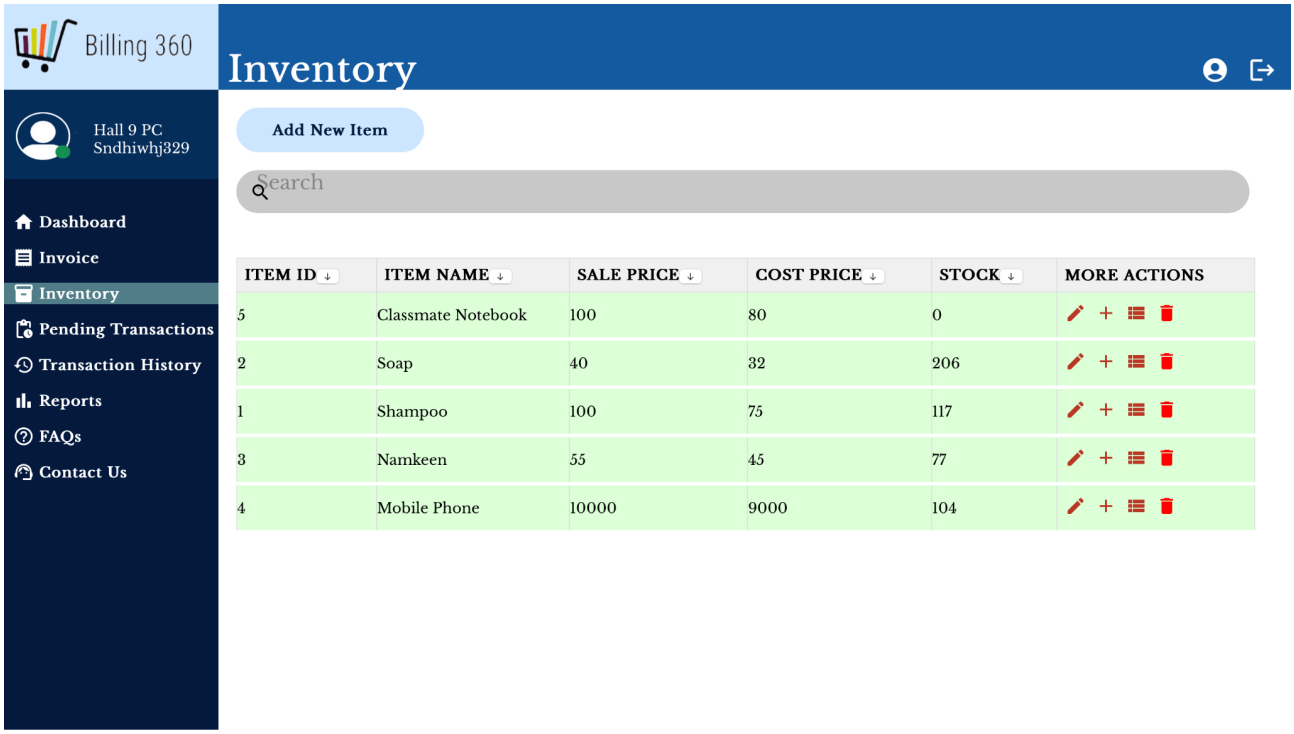
Test #2

Details : To verify the feature of adding a new item in the inventory.

Adding a new product



Display of the new product in the inventory page verifying the addition in database.

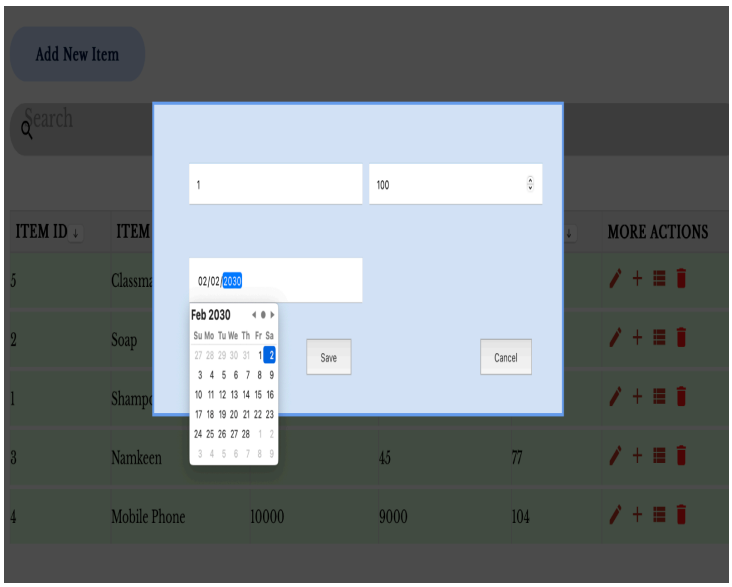


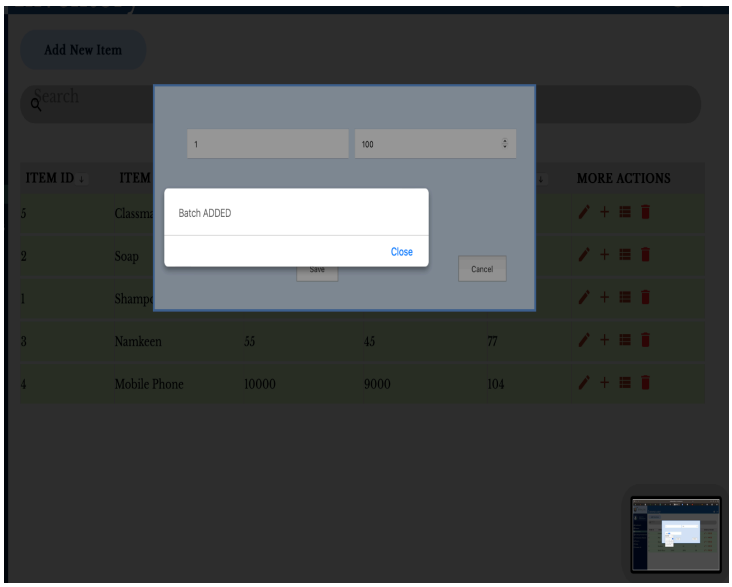
Result : The new item is successfully added into the inventory.

Test #3

Details : To verify the feature of adding a batch for an item in the inventory.

Adding a new batch



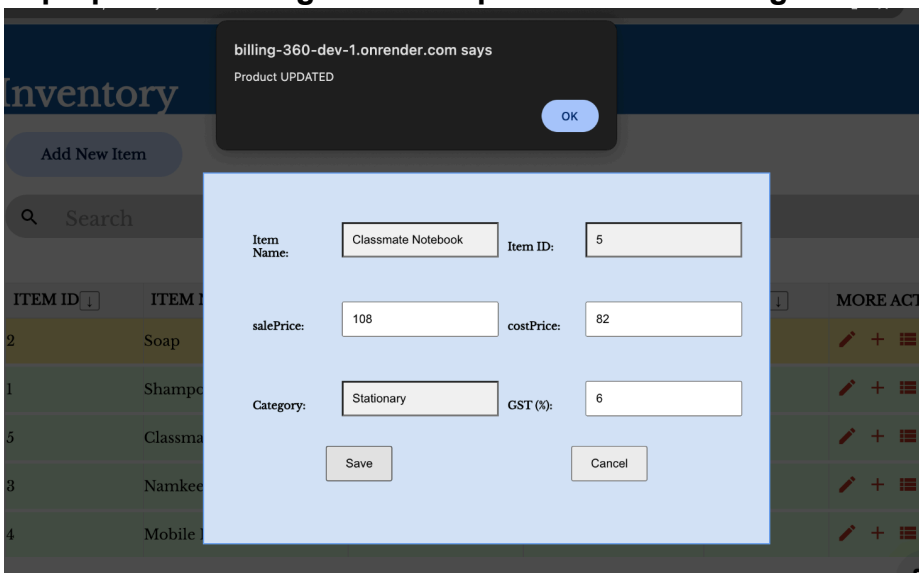


Result : Batch is added successfully with a pop-up box confirming that.

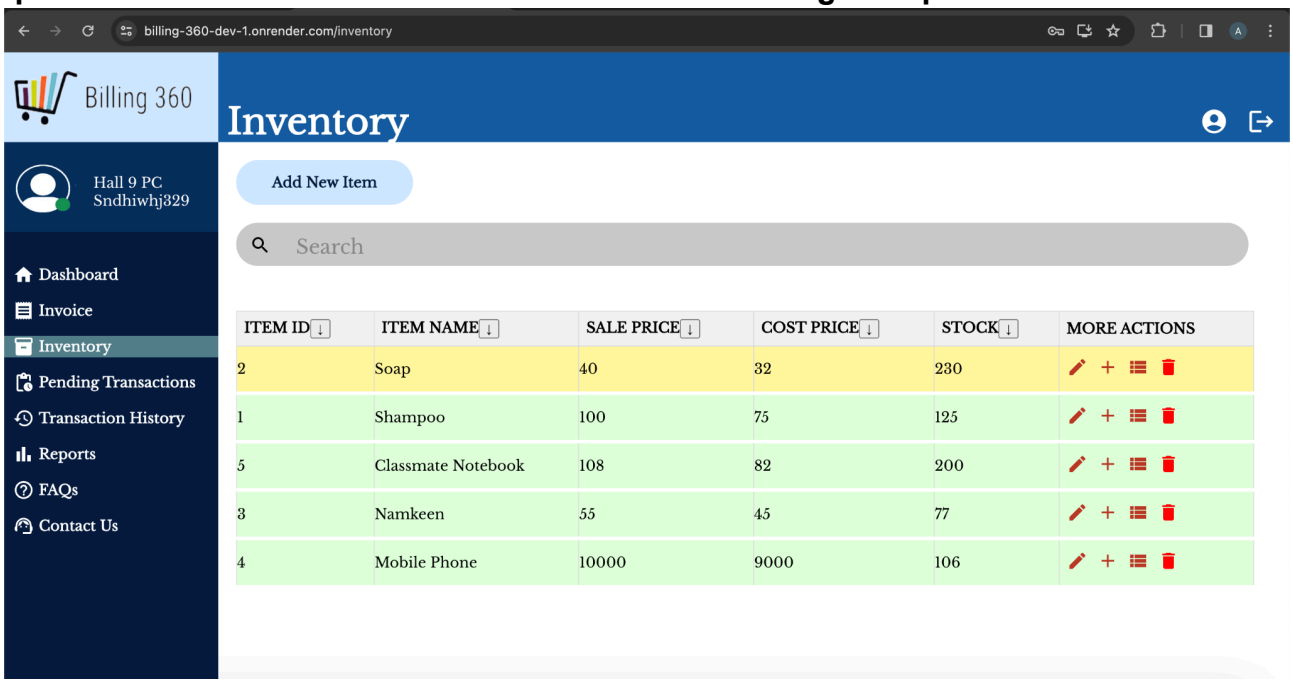
Test #4

Details : To verify the feature of updating the details of an item in the inventory.

Pop-up box showing Product Updated after clicking on save button.



Updated information received from backend confirming the update in database.

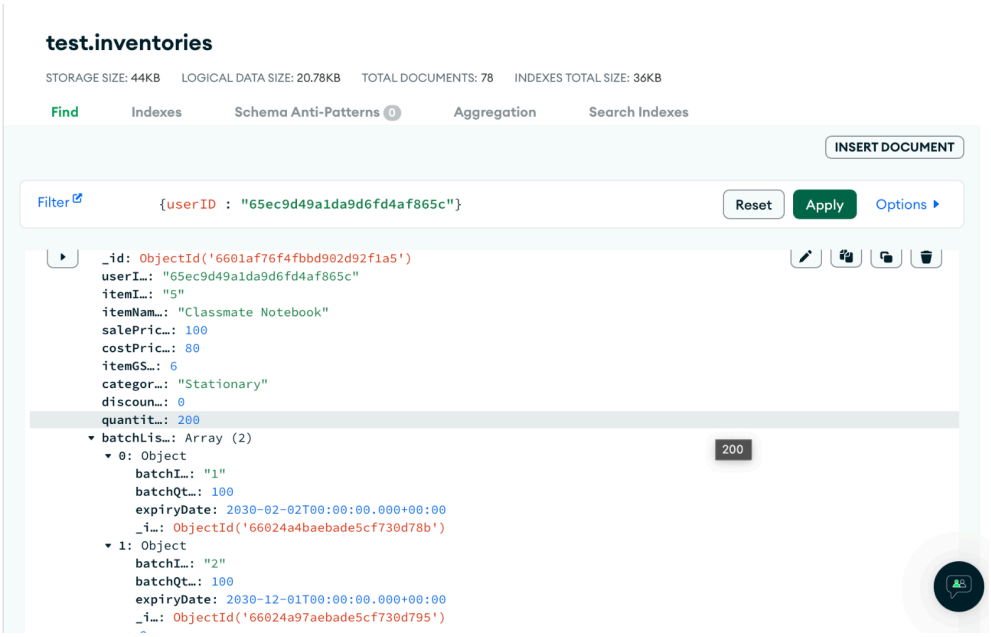


Result : The information is successfully updated via edit item feature.

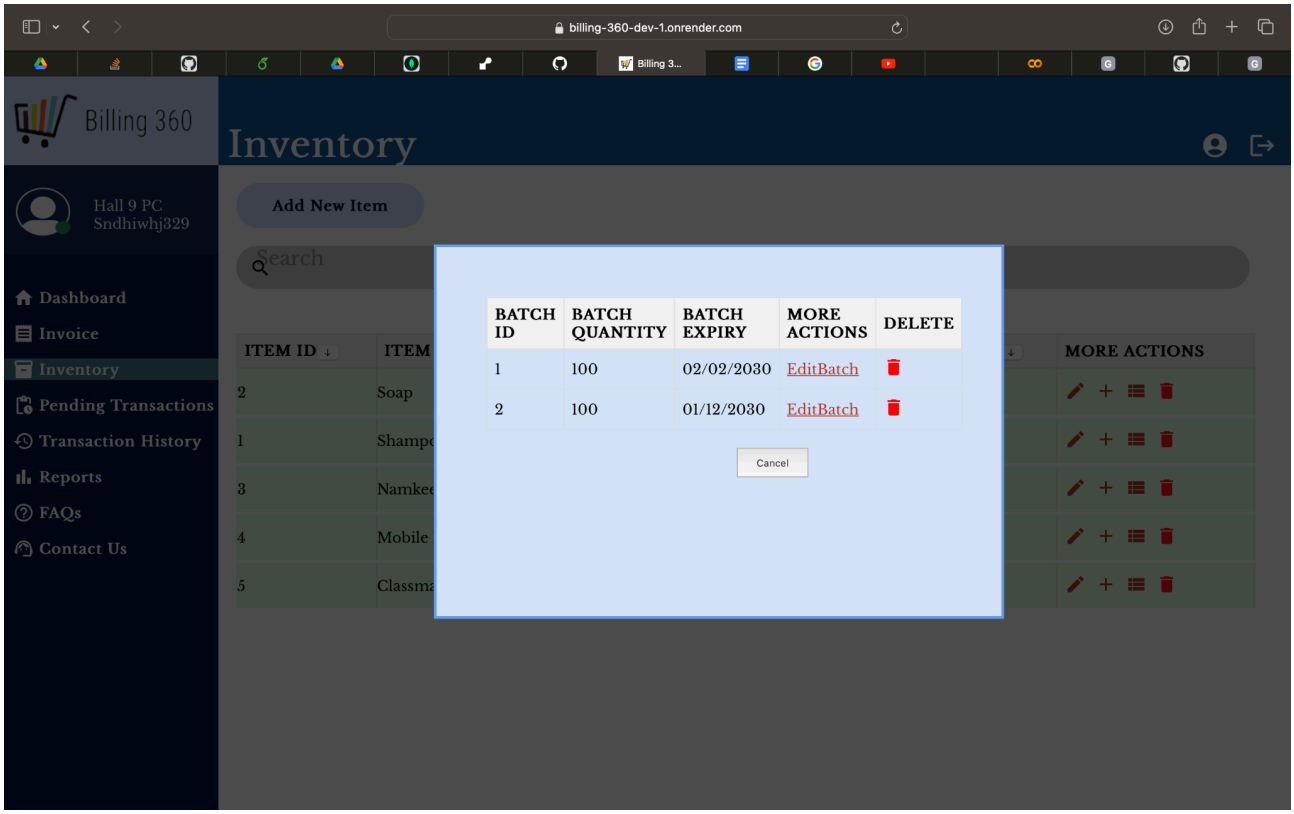
Test #5

Details : To verify the feature of viewing the details of a batch of a particular item.

Information of the batches of a particular item in the database.



View Batch Dialog showing the correct data of the batches.

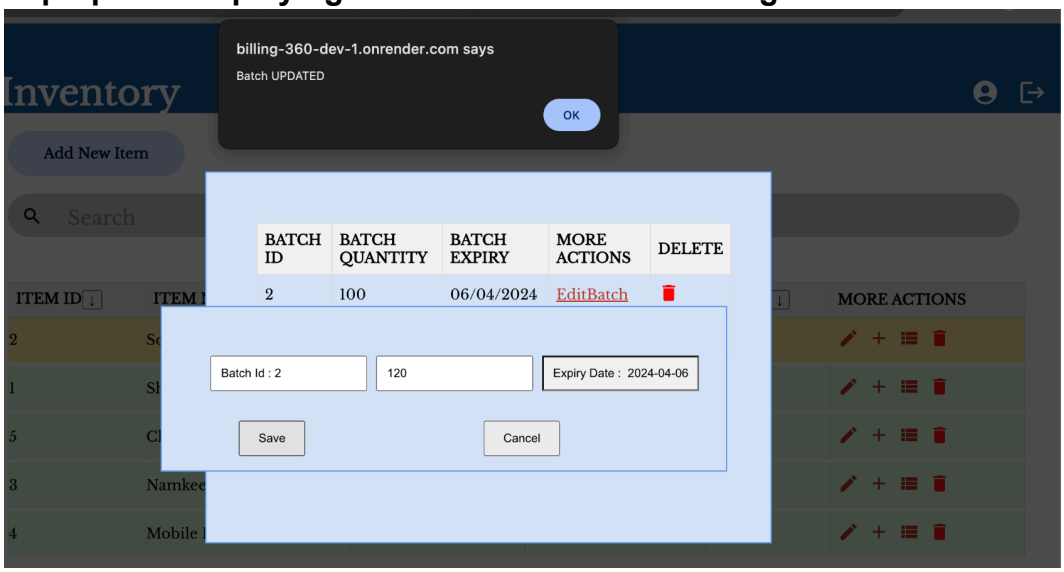


Result : The data of the view Batch dialog in frontend and corresponding data in database are matching.

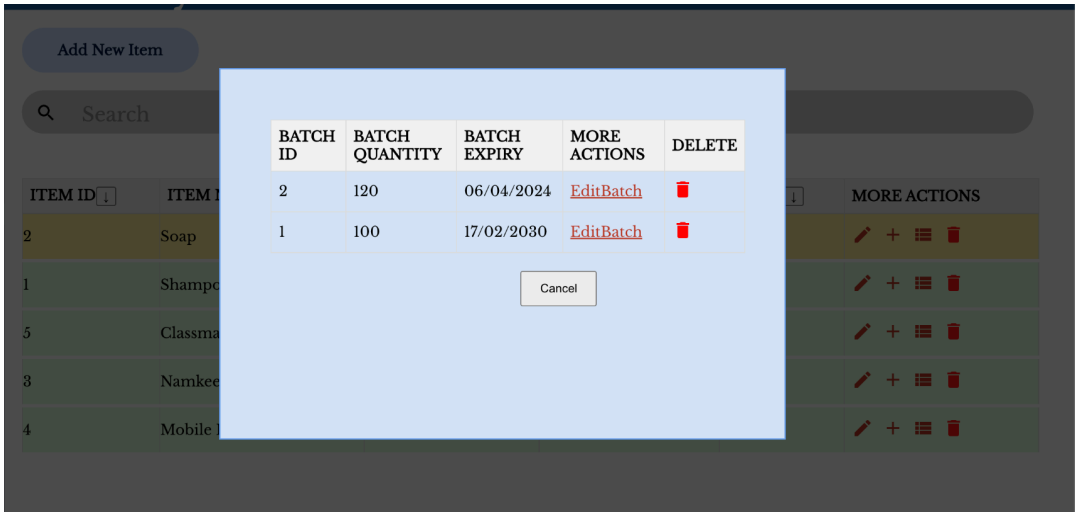
Test #6

Details : To verify the feature of editing the details of a batch of a particular item.

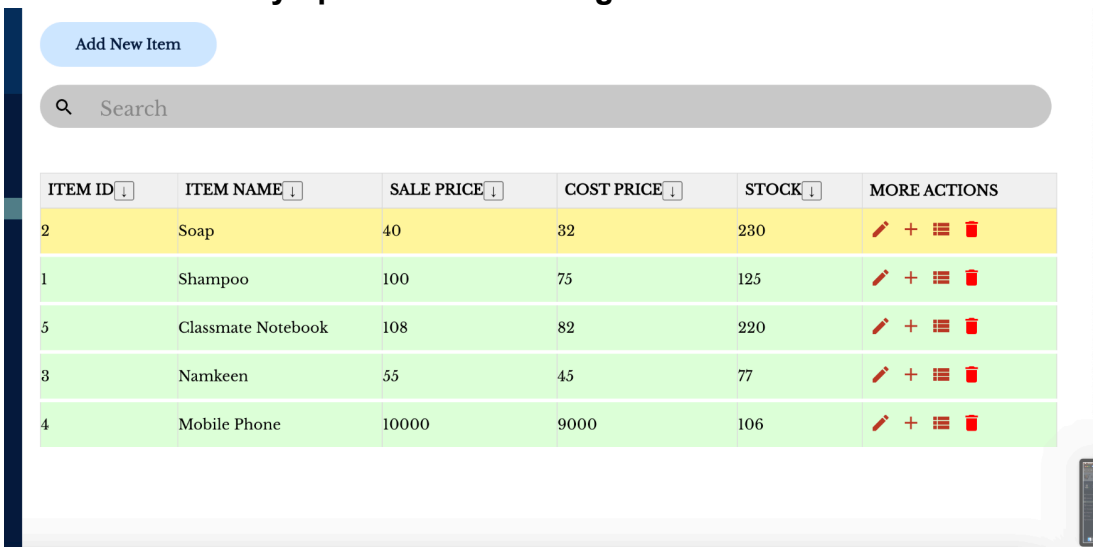
Pop-up box displaying Batch UPDATED on clicking the save button.



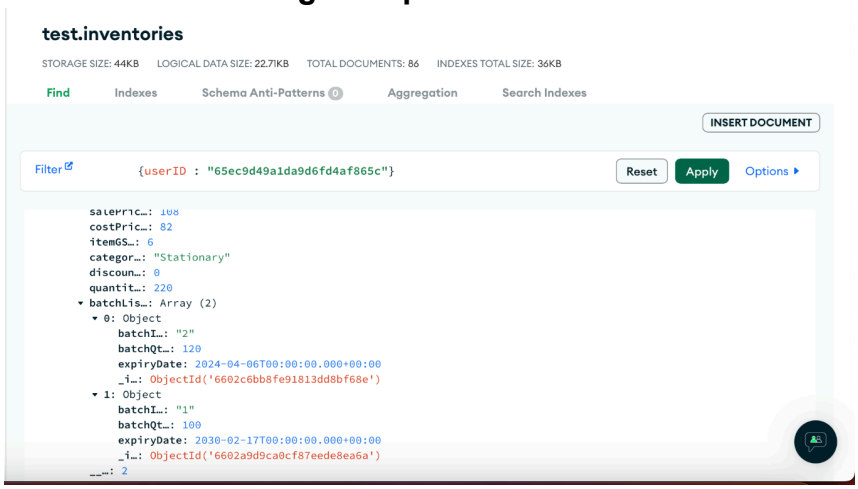
Information correctly updated in the view Batch Dialog.



Stock successfully updated after editing the details of batch.



Database confirming the update.

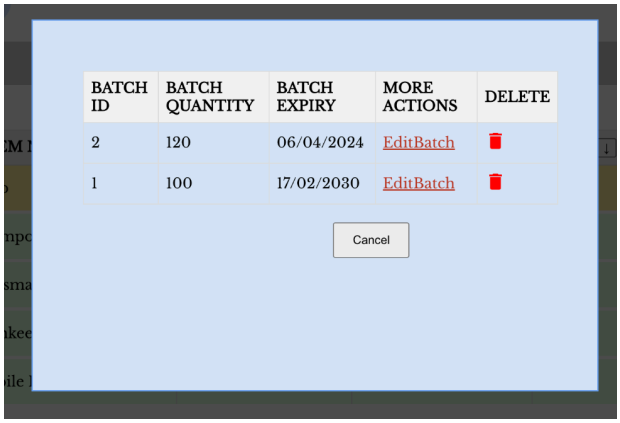


Result : Batch details are successfully updated.

Test #7

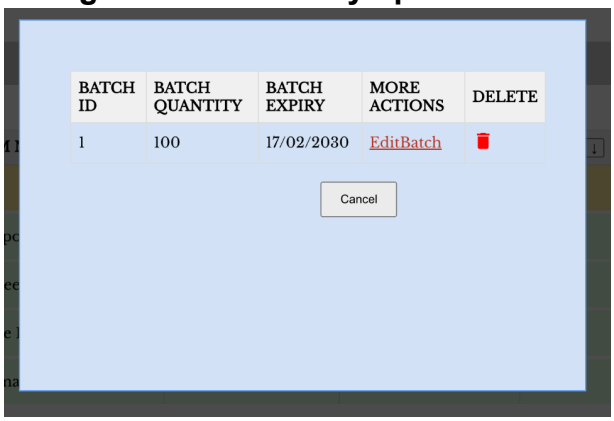
Details : To verify the feature of deleting a batch of a particular item.

Current Batch data

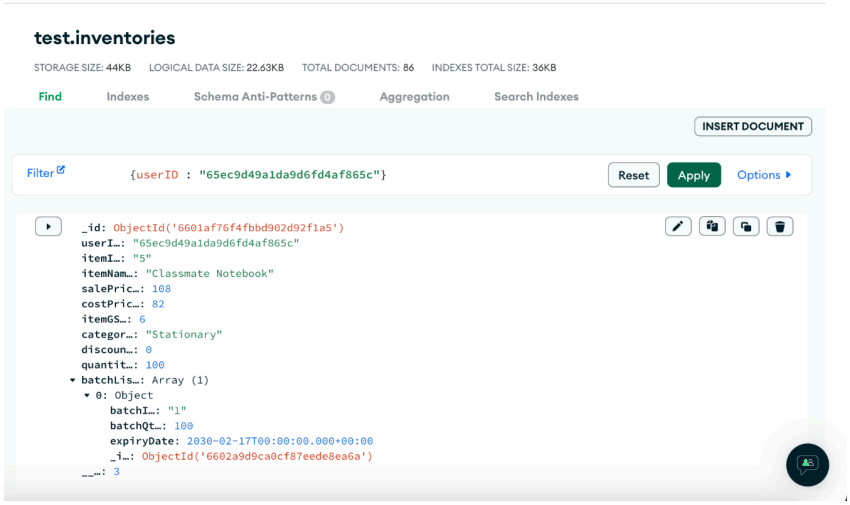


Deleting Batch with batch Id 2.

Dialog box successfully updated.



Both the batch data and total stock successfully updated in the database.

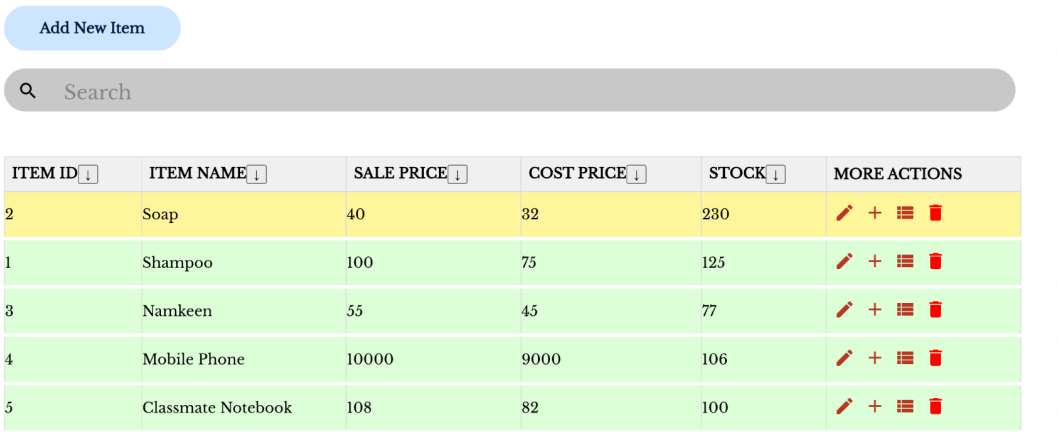


Result : The batch deletion is verified successfully.

Test #8

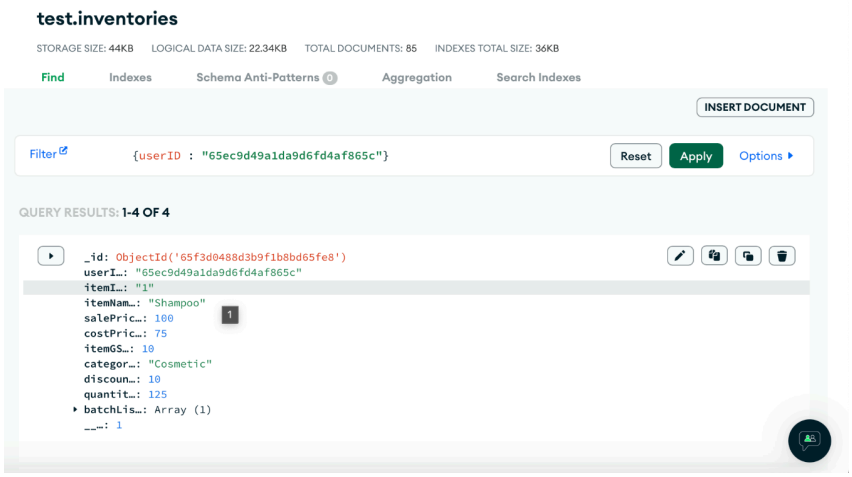
Details : To verify the feature of deleting an item from inventory.

Initial state before delete operation on “Classmate Notebook”.



State after pressing delete icon in the row corresponding to “Classmate Notebook”

Database successfully updated.



Result : Successful verification of deleting an item feature from inventory.

3 Pending Transactions

Module Details: Ensure the seamless functionality of the invoicing system across various modules, including invoice creation, automatic email notifications for credit additions and payments, and updating customers via email. Additionally, validate the proper api calls to edit customer information, manual debit/credit additions, invoice generation and transaction history updates

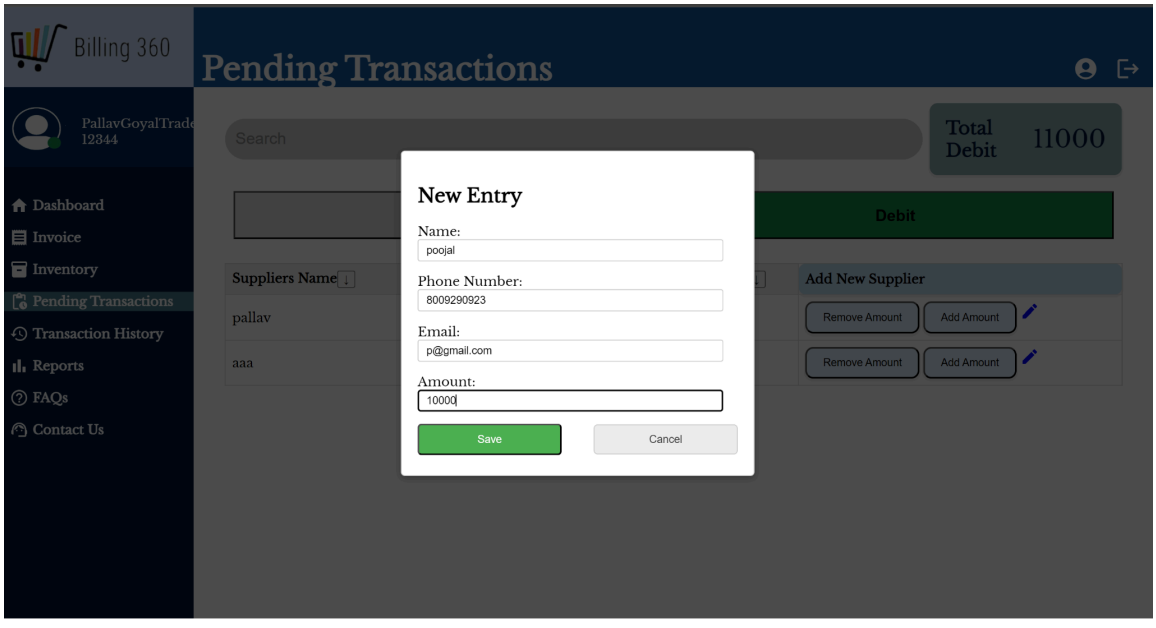
Test Owner : Poojal Katiyar

Test Date : 26/03/2023

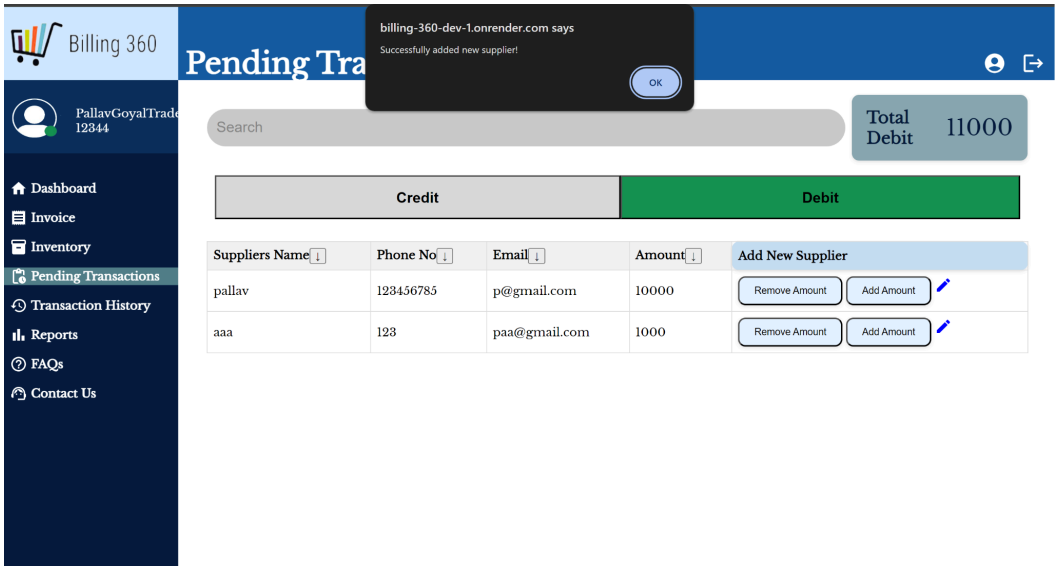
Test Results : In this test we validated the successful integration of backend and frontend components of Add new customer/Supplier, Update entry ,Clear dues,Add as credit ,generating invoices on clear dues/Add as Credit and sending it to the customer via email.Apart from this successfully tested that entered amount for clear dues does not exceed the due amount or negative amount is entered.Throughout the testing process, we also scrutinized the redirection of users to the correct pages upon the completion of these actions.

Test #1

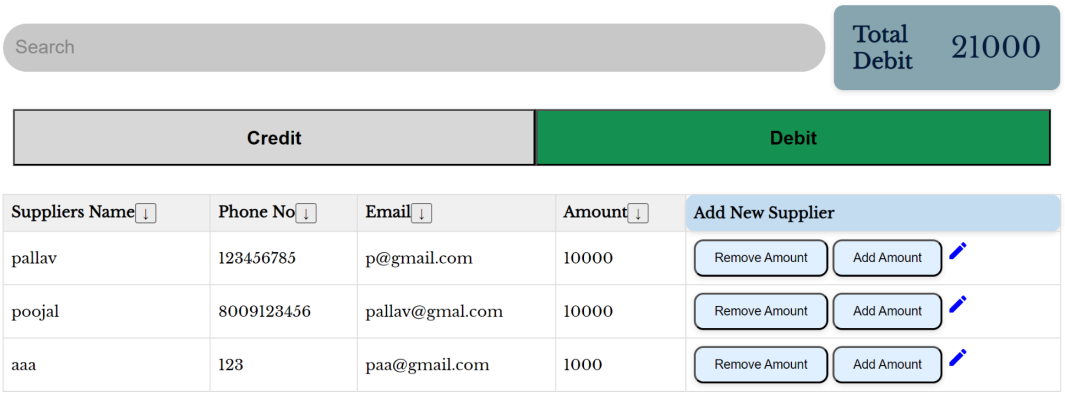
Details : To verify the proper functioning of Add New Supplier in the Debit tab.



Result: Add new Supplier dialog box open in which details are entered.

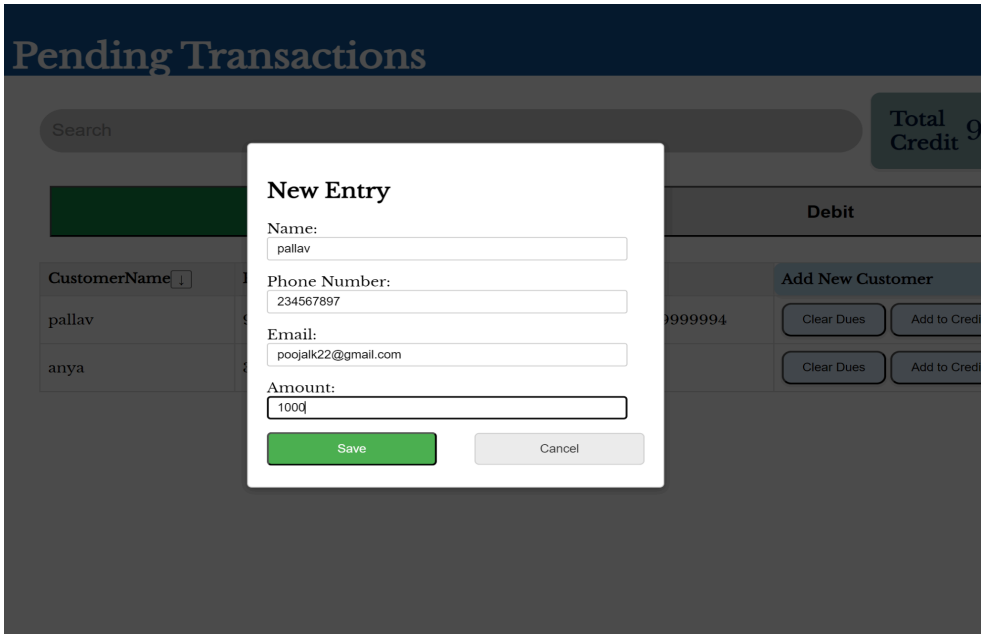


Result:Suppliers name is updated in the table

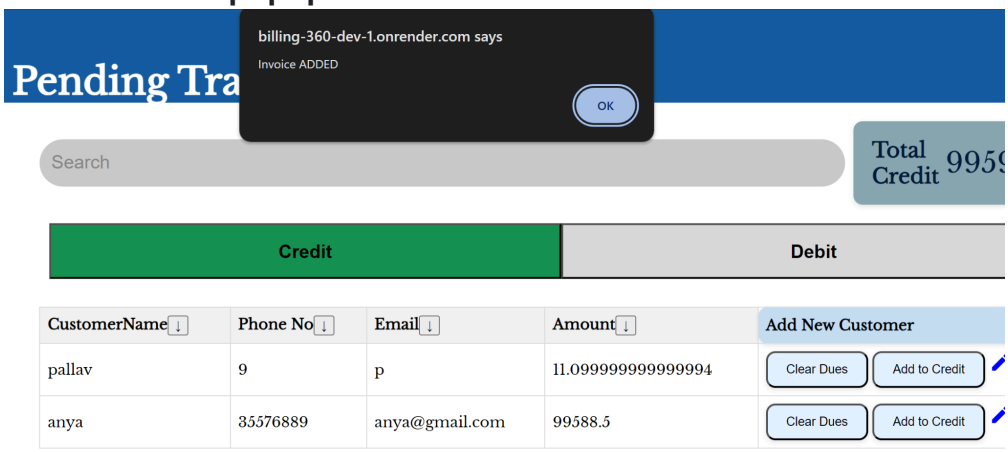


Test #2

Details : On Adding New Customer ,Invoice is generated regarding how much amount is due by customer, new customer is added in the database and pdf is mailed to Customer's Email.



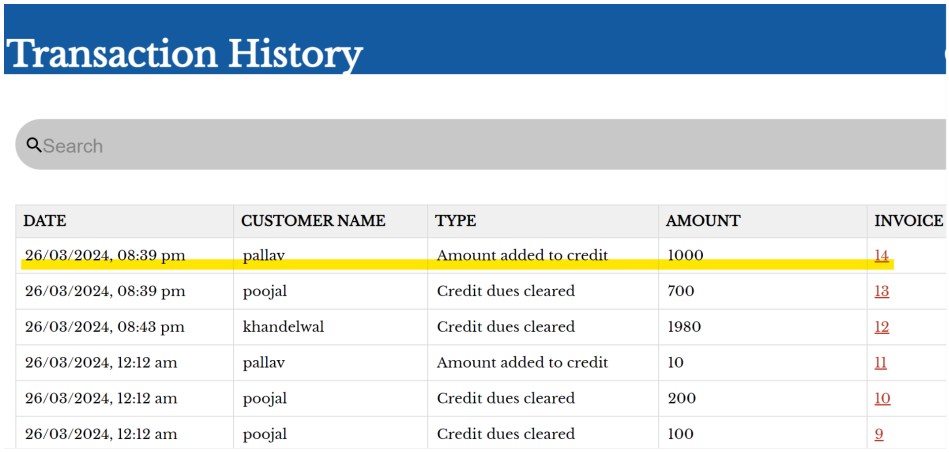
Invoice Added popup Box



Database of Customers is updated


```
_id: ObjectId('6602ef627183aa864feccbe8')
userID: "65f3507fc86c4fb597696653"
name: "pallav"
phoneNo: "234567897"
email: "poojalk22@gmail.com"
creditAmount: 1000
▼ invoiceList: Array (1)
  0: ObjectId('6602ef637183aa864feccbeb')
__v: 1
```

Invoice is generated and added to transaction history as “Amount added to Credit”

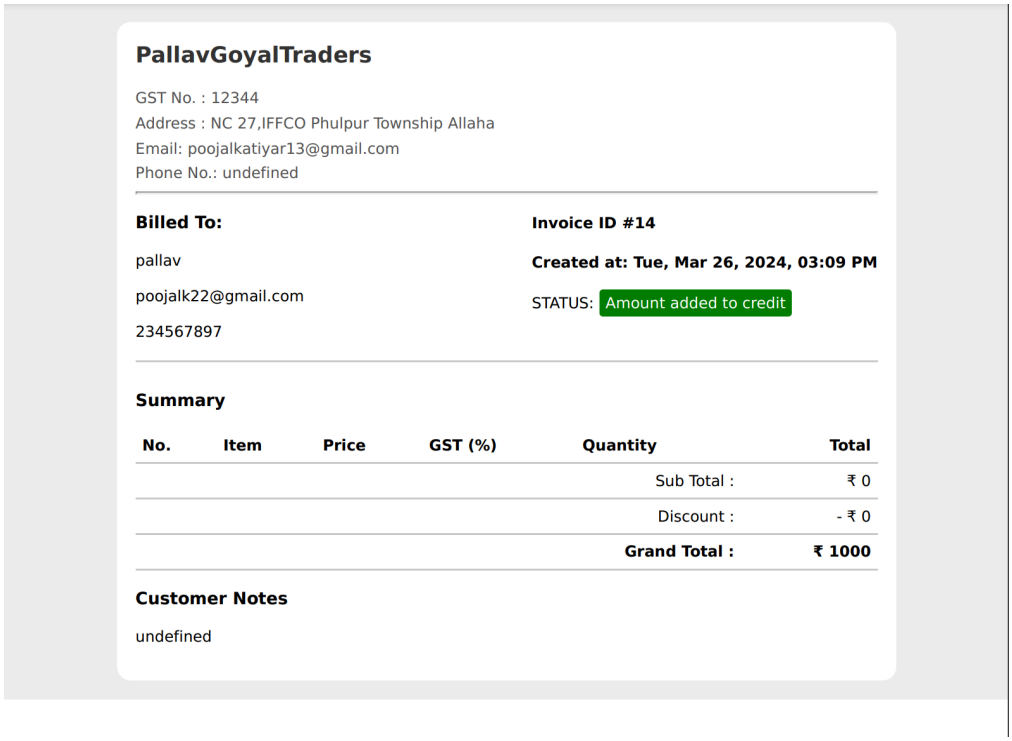


Transaction History

Search

DATE	CUSTOMER NAME	TYPE	AMOUNT	INVOICE
26/03/2024, 08:39 pm	pallav	Amount added to credit	1000	14
26/03/2024, 08:39 pm	poojal	Credit dues cleared	700	13
26/03/2024, 08:43 pm	khandelwal	Credit dues cleared	1980	12
26/03/2024, 12:12 am	pallav	Amount added to credit	10	11
26/03/2024, 12:12 am	poojal	Credit dues cleared	200	10
26/03/2024, 12:12 am	poojal	Credit dues cleared	100	9

Invoice PDF



PallavGoyalTraders

GST No. : 12344
Address : NC 27,IFFCO Phulpur Township Allaha
Email: poojalkatiyar13@gmail.com
Phone No.: undefined

Billed To:

pallav

poojalk22@gmail.com

234567897

Invoice ID #14

Created at: Tue, Mar 26, 2024, 03:09 PM

STATUS: Amount added to credit

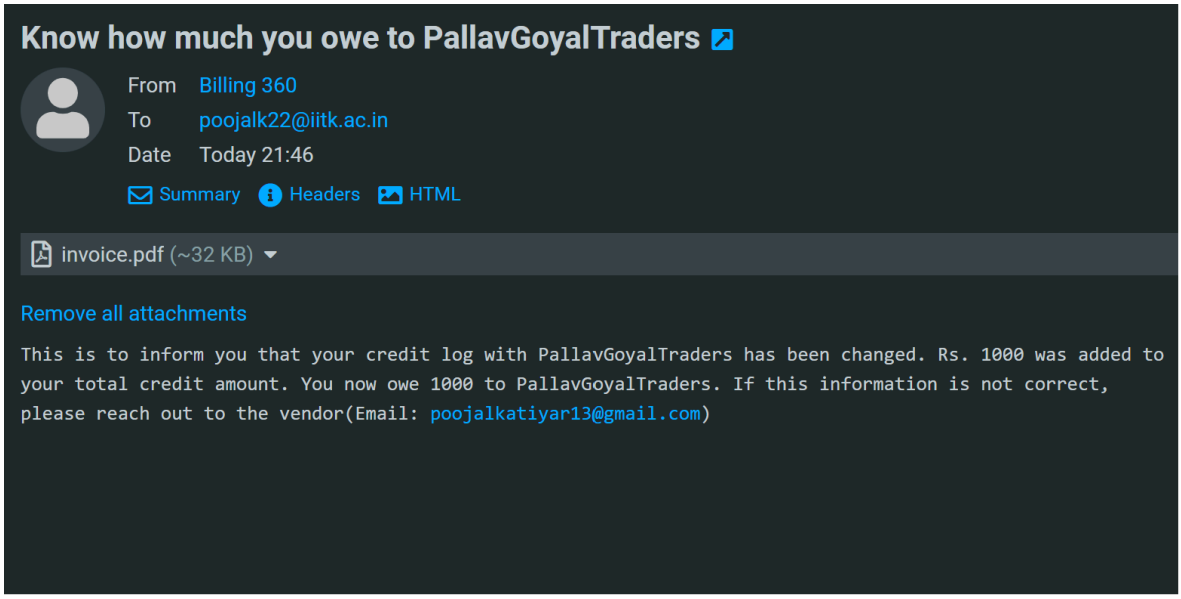
Summary

No.	Item	Price	GST (%)	Quantity	Total
				Sub Total :	₹ 0
				Discount :	- ₹ 0
				Grand Total :	**₹ 1000**

Customer Notes

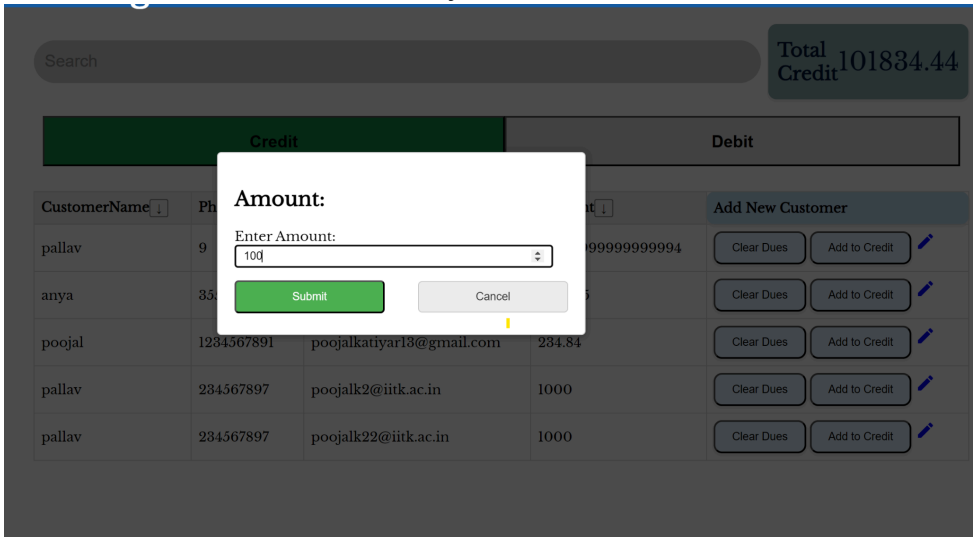
undefined

PDF of invoice sent to customers’s email

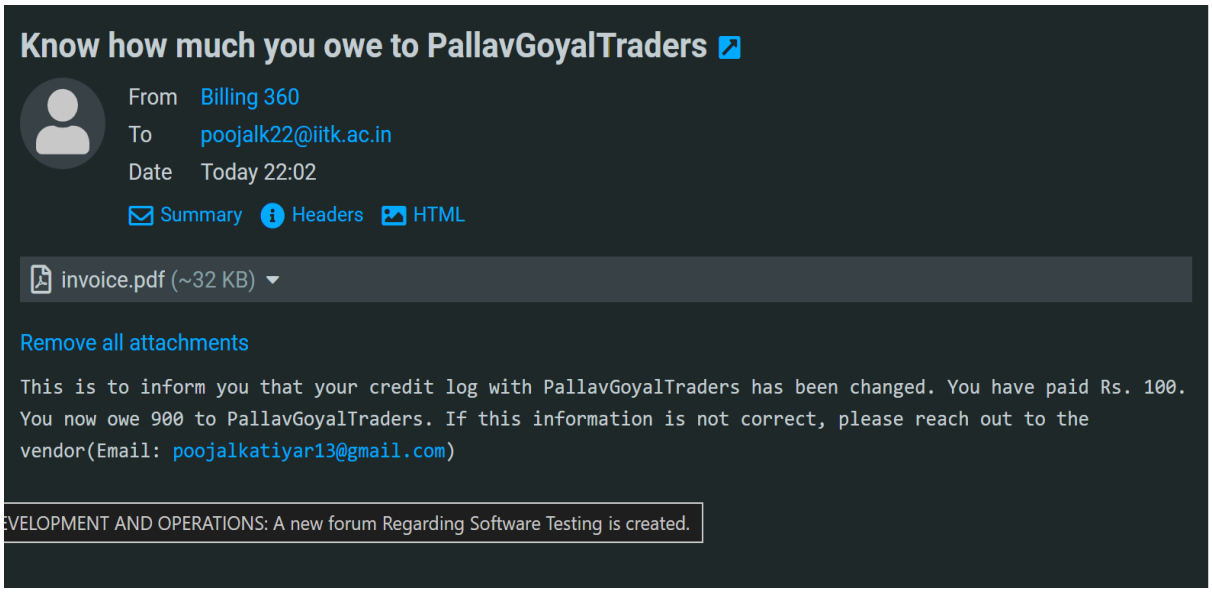


Test #3

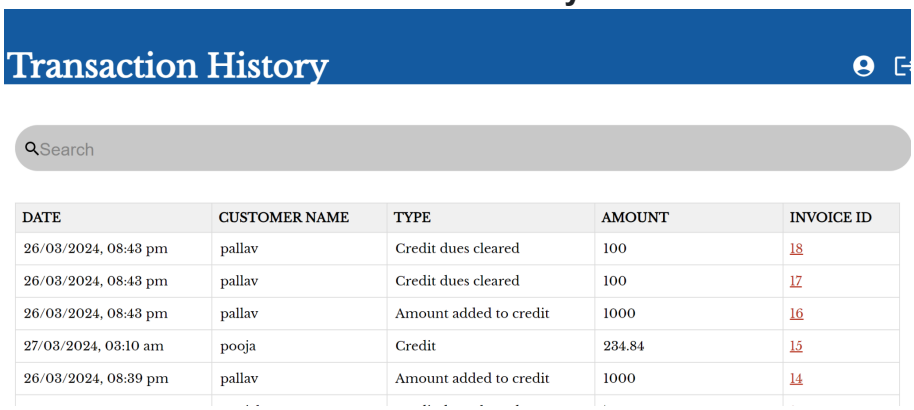
Details : Clear Dues functionality



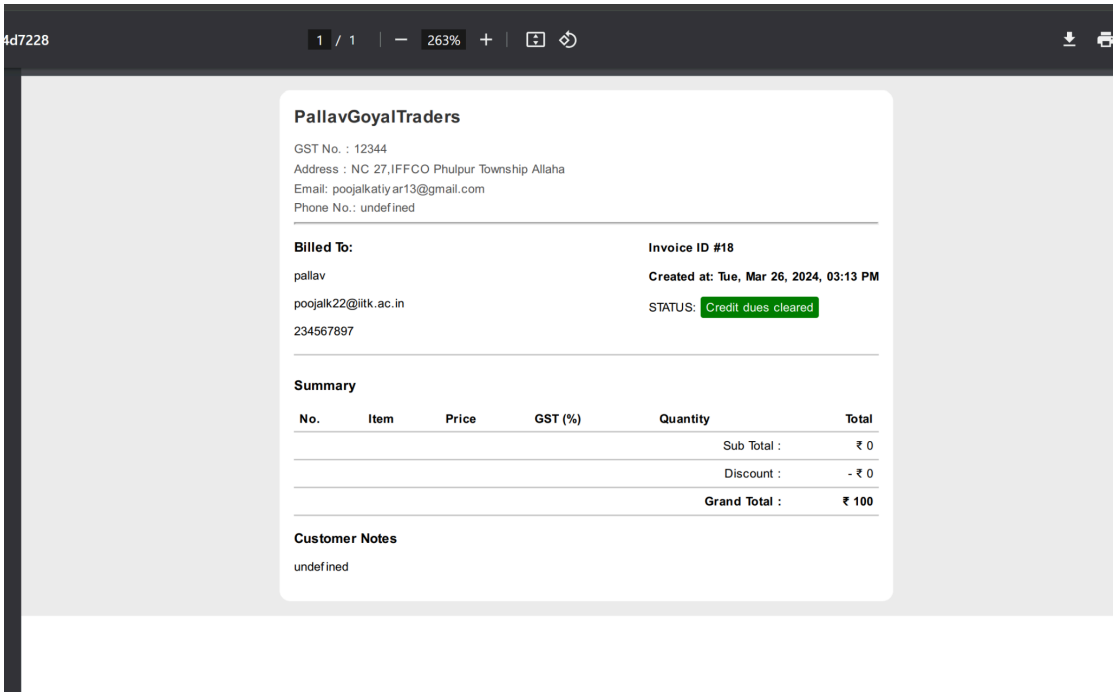
Mail Send to Customer regarding Amount paid to the Shopkeeper



Invoice Added in Transactions History



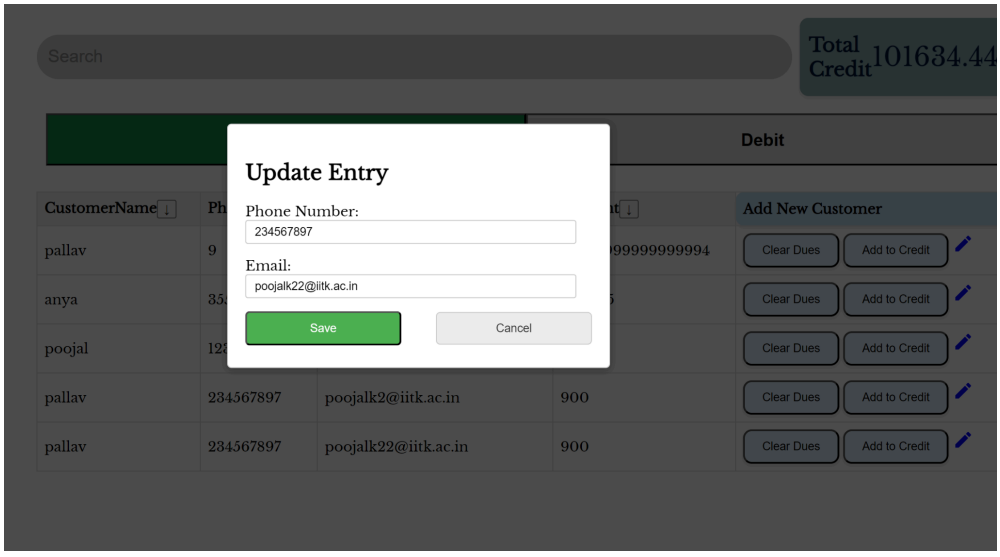
Invoice PDF

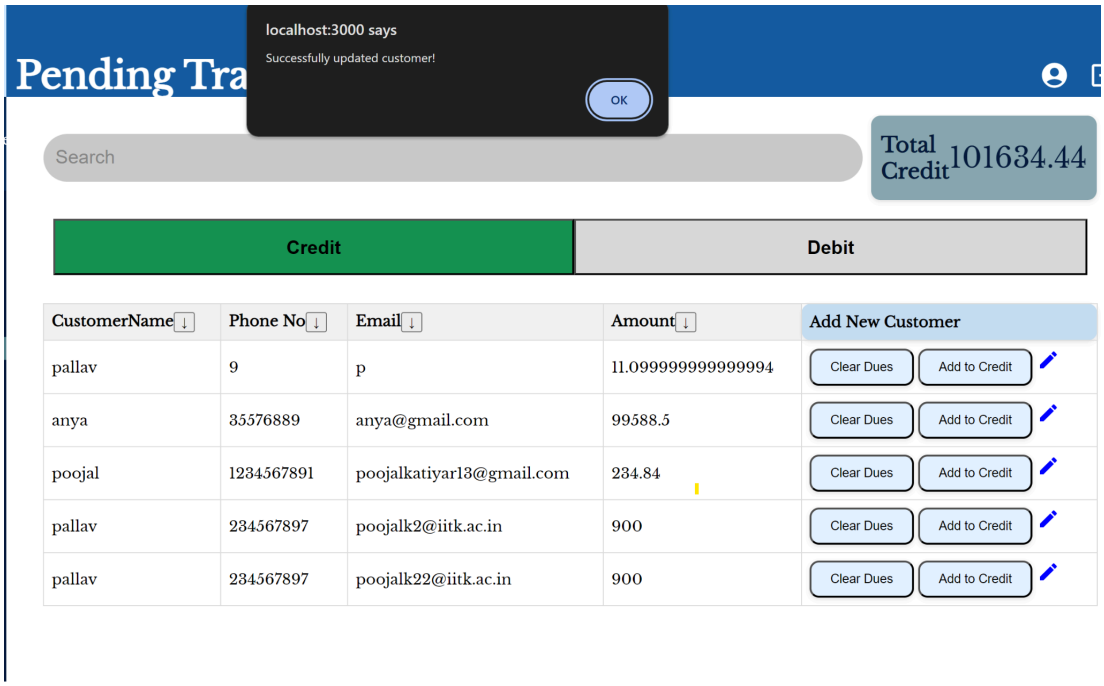


Result: Dues cleared by the customer are updated in credit and the customer is informed about the amount paid and transaction history is updated.

Test #4

Details : Edit Customer's Details





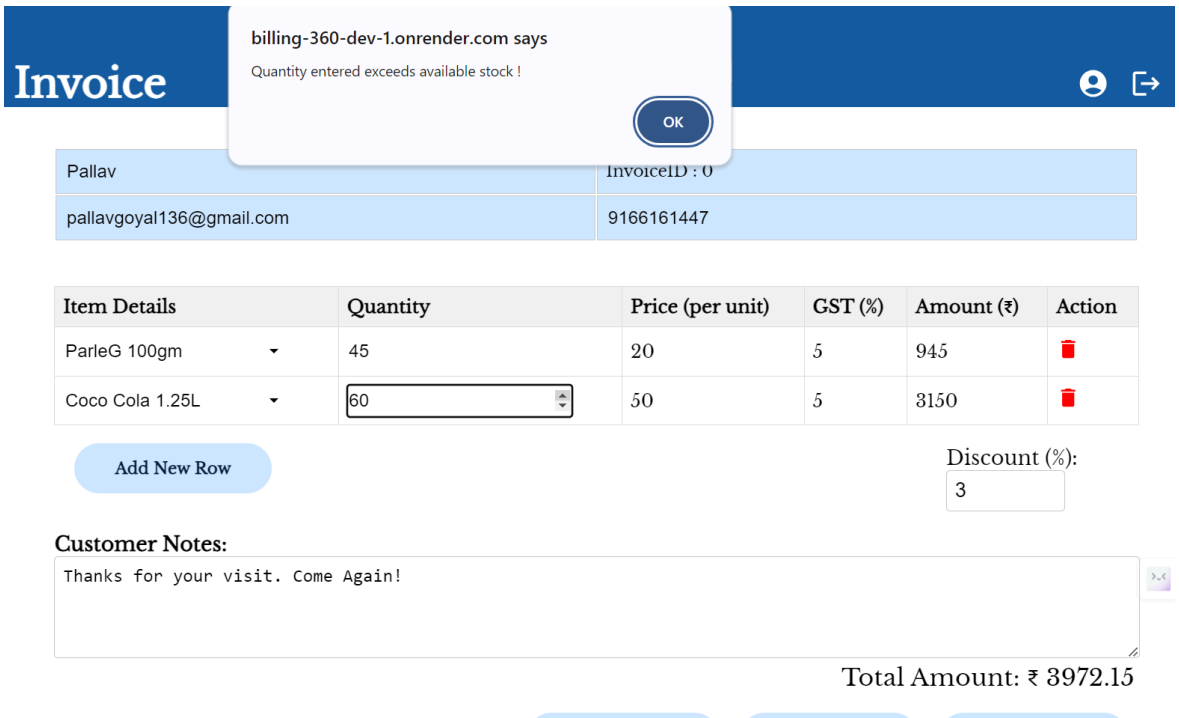
Result: pop up Box showing “Successfully updated customer!”

4. Invoice

Module Details: This model ensures seamless functionality of the invoicing system across various modules including invoice creation, automatic email notifications after invoice creation,transaction history updates,dashboard updates,preview bill and generated bill.After bill generation, the system automatically updates inventory levels batch-wise, alongside transaction history and customer databases.It enforces restrictions to ensure accuracy and efficiency, such as preventing users from entering quantities exceeding available inventory and allowing only one item per row in invoices. Additionally, it includes validating the redirection to respective pages upon completion of these actions.
Test Owner: Pallav Goyal
Test Date: 26/03/2024
Test Results: In this test we validated the successful integration of backend and frontend components of dashboard, pending transactions, transaction history mailing and viewing pdf feature.We also validated that database of invoice and customer is updated on generate bill.Here we are integrating inventory api, customer api, pdf generation api, mailing api and invoice api using frontend.

Test #1

Details :Quantity entered in the invoice greater than available in the inventory

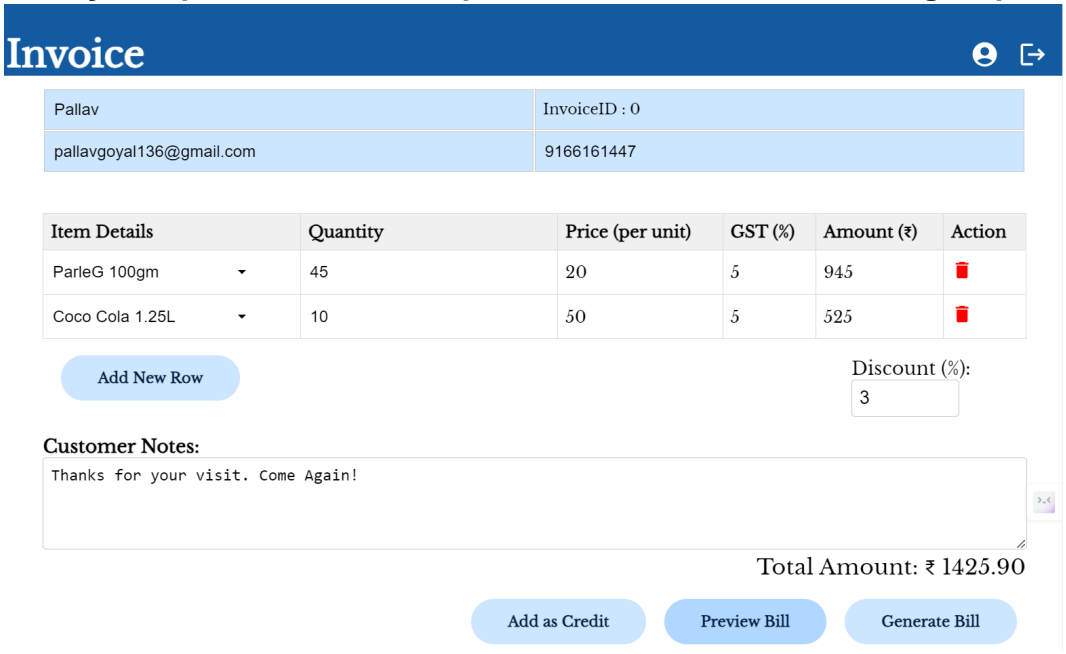


Result: Pop up box that displays “Quantity entered exceeds available stock!”

Test #2

Details :Creating an Invoice and generating the bill and sending it via email of the customer.

Firstly fill up all the details required in the invoice and setting as paid.



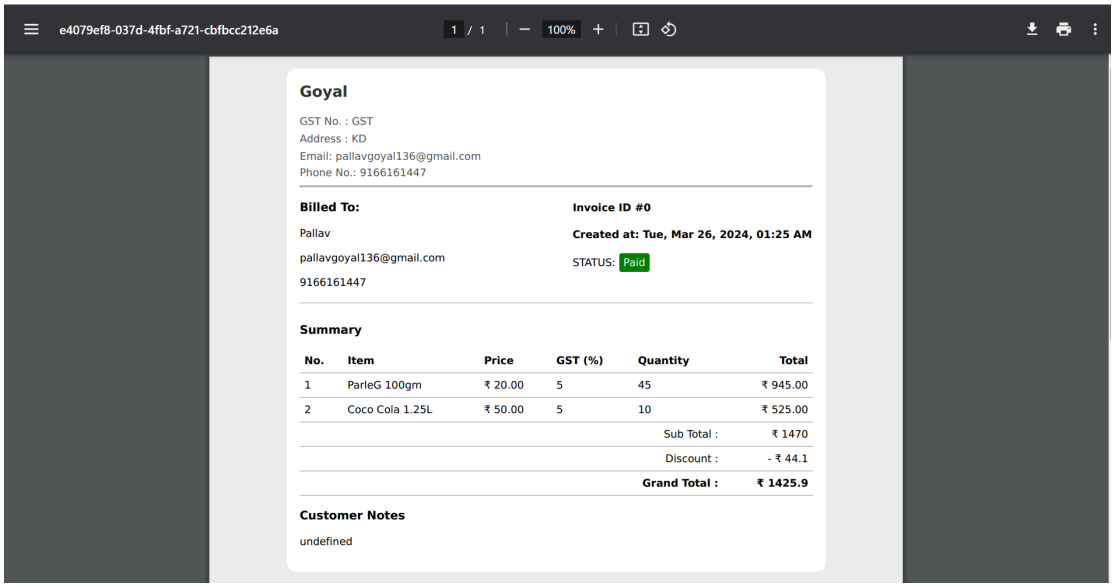
Invoice added to database successfully on generating bill

```
_id: ObjectId('6601d71dbff0240b92f7c0b4')
userID: "65f45c105fcad1b212b4bbe3"
invoiceID: 0
customerName: "Pallav"
phoneNo: "9166161447"
customerEmail: "pallavgoyal136@gmail.com"
totalAmount: 1425.9
notes: "Thanks for your visit. Come Again!"
paymentMode: "Paid"
discount: 3
▼ itemList: Array (2)
  ▸ 0: Object
  ▸ 1: Object
createdAt: 2024-03-26T01:27:20.637+00:00
totalCostPrice: 1075
totalSales: 1400
```

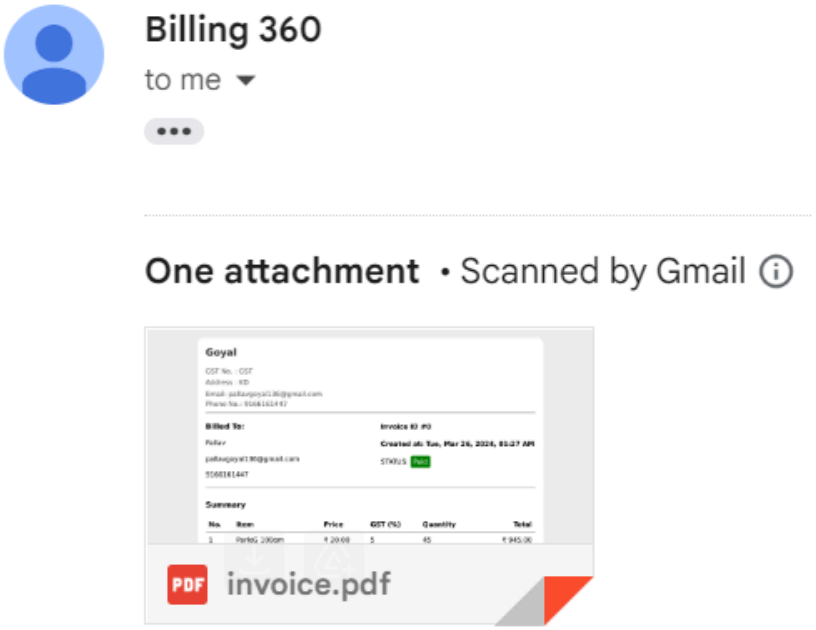
New customer of shopkeeper added to the customer database

```
_id: ObjectId('6601d71ebff0240b92f7c0bc')
userID: "65f45c105fcad1b212b4bbe3"
name: "Pallav"
phoneNo: "9166161447"
email: "pallavgoyal136@gmail.com"
creditAmount: 0
▼ invoiceList: Array (1)
    0: ObjectId('6601d71dbff0240b92f7c0b4')
__v: 0
```

Result:PDF gets shown on clicking Preview Bill as generate pdf api gets successfully completed



Result: Invoice mailed on the customer's email.

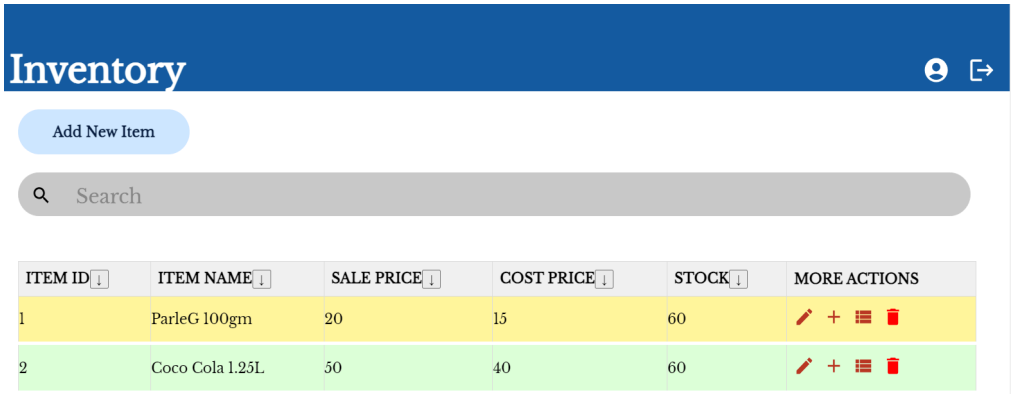


Test #3

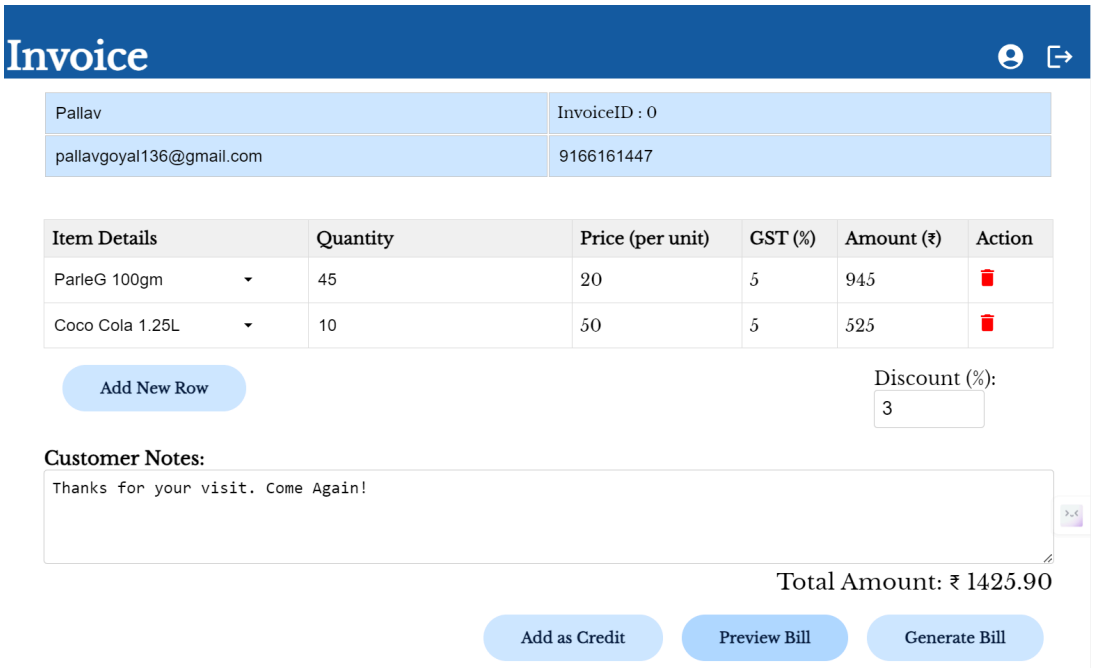
Details :Reduction in the items in inventory on creating invoice.

Steps are shown below:

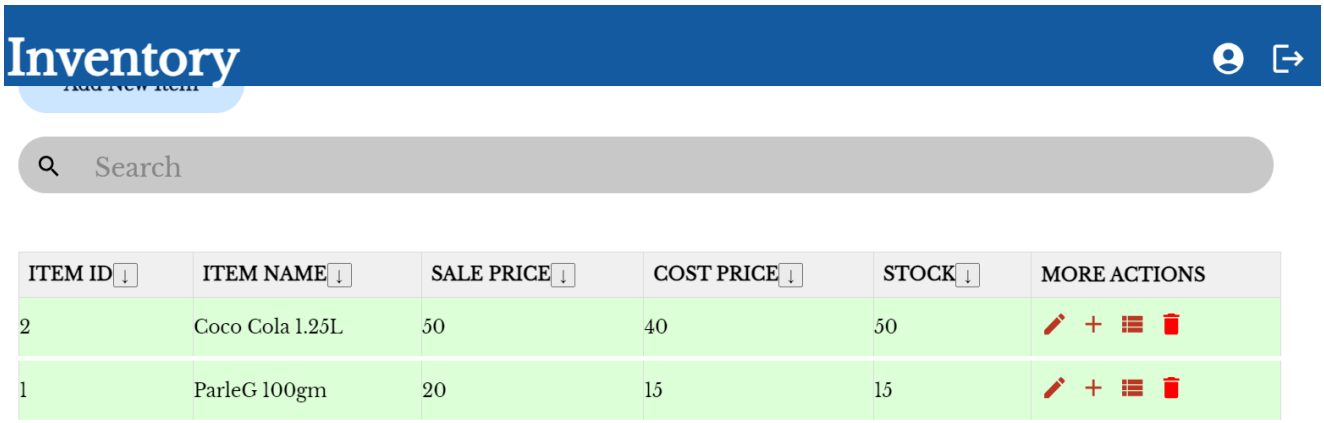
1.State of inventory before generating bill



2.Invoice creation



3.State of inventory after generating bill showing reduced items



Result: On creating invoice,items are reduced in inventory.

Test #4

Details: Items of earlier batches are reduced first from the inventory upon creating invoice.

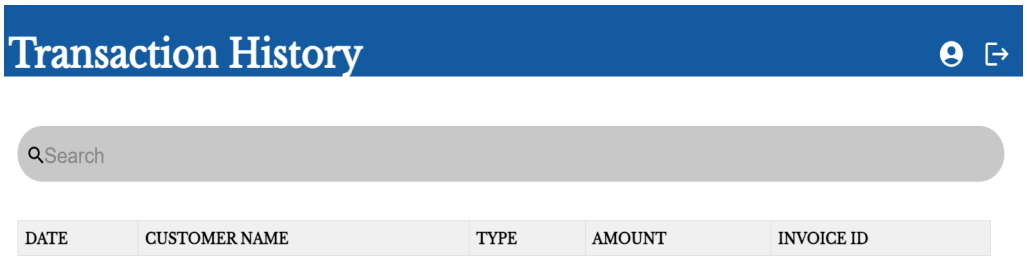
State of batches of ParleG before generating bill

State of batches of ParleG after generating bill showing items of earlier batch finished earlier

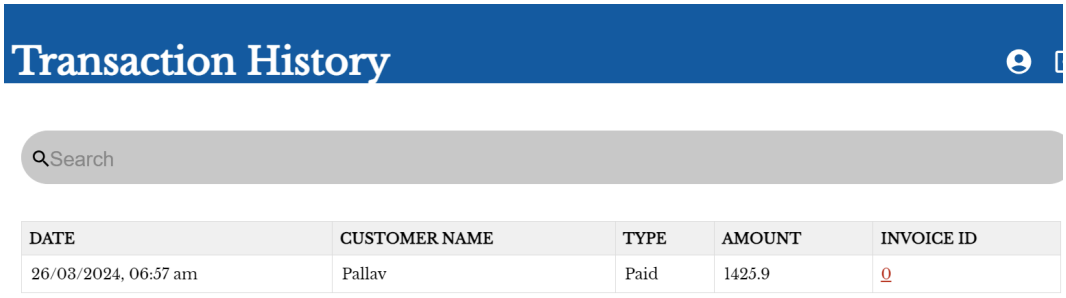
Test #5

Details Changes in Transaction history before and after generation of invoice and feature to view the invoice from transaction history.

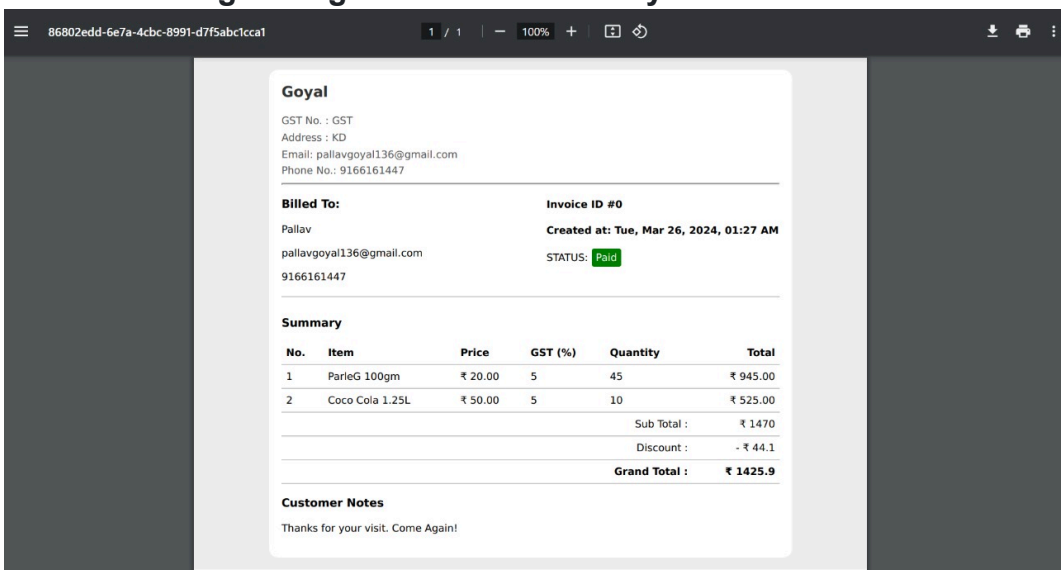
State of transaction history before generating bill



State of transaction history after generating bill



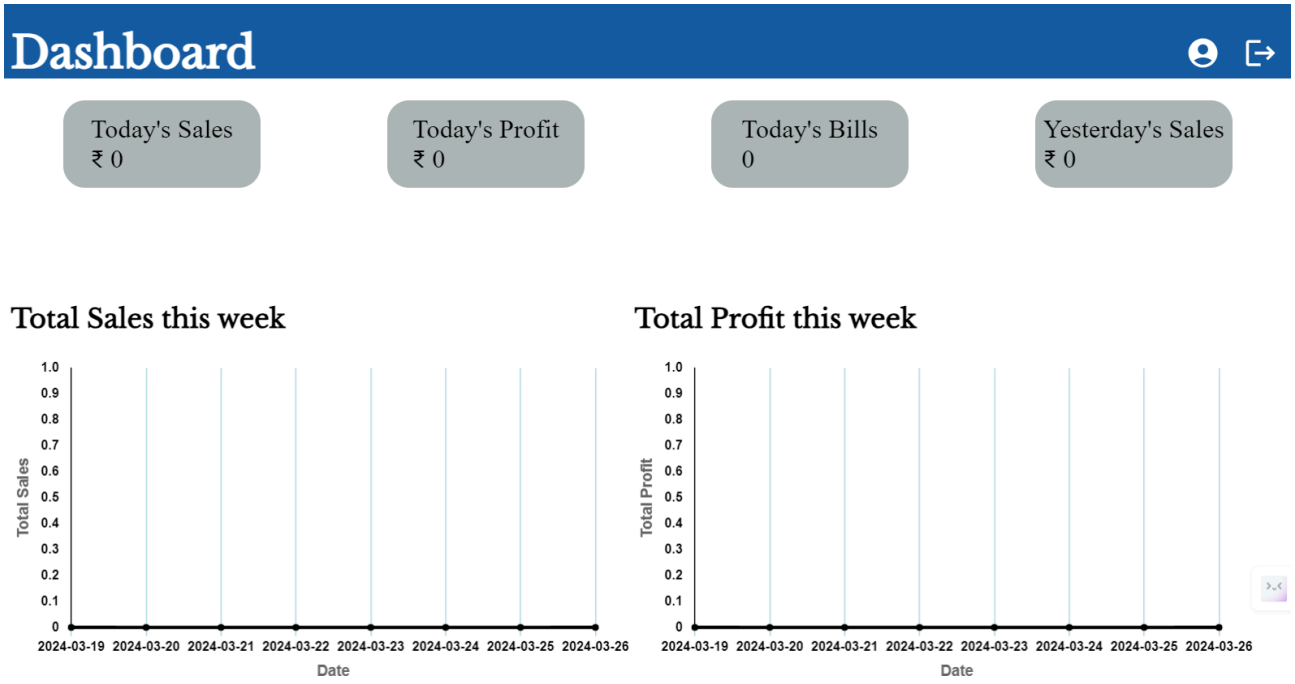
Invoice viewing through transaction history



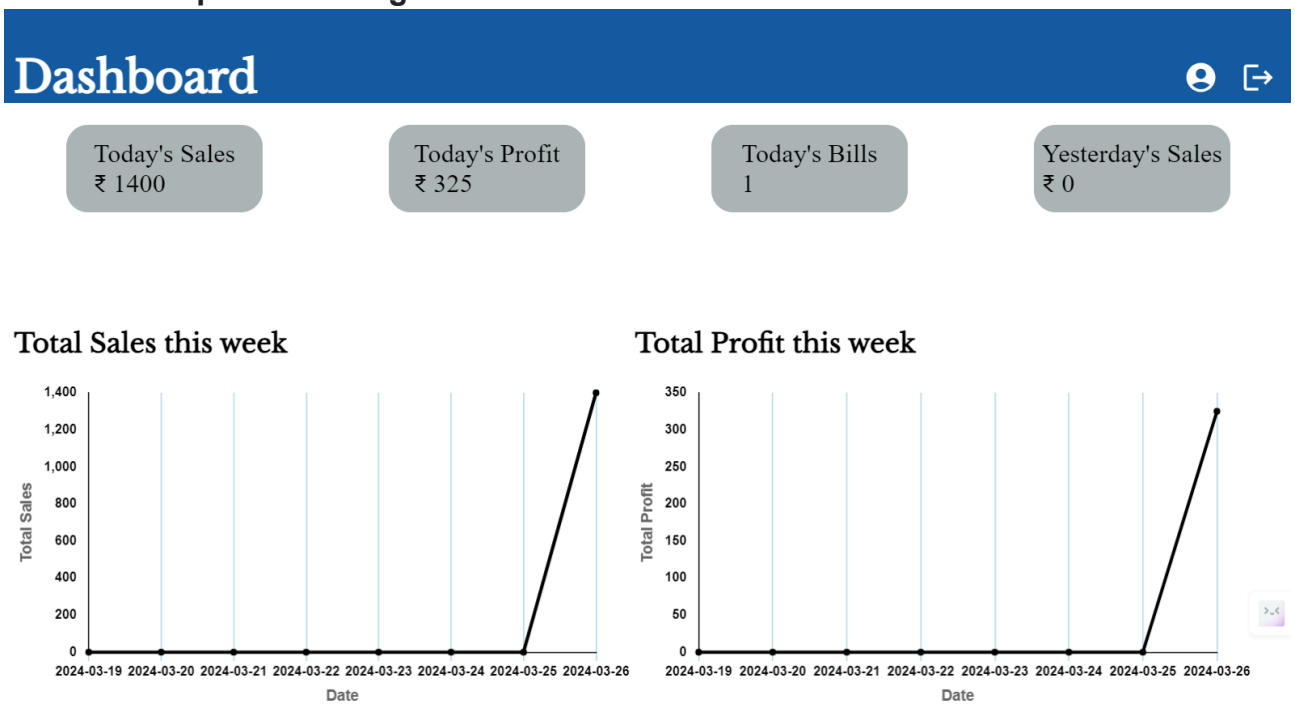
Test #6

Details Changes in Dashboard before and after generation of invoice

Dashboard before generating bill



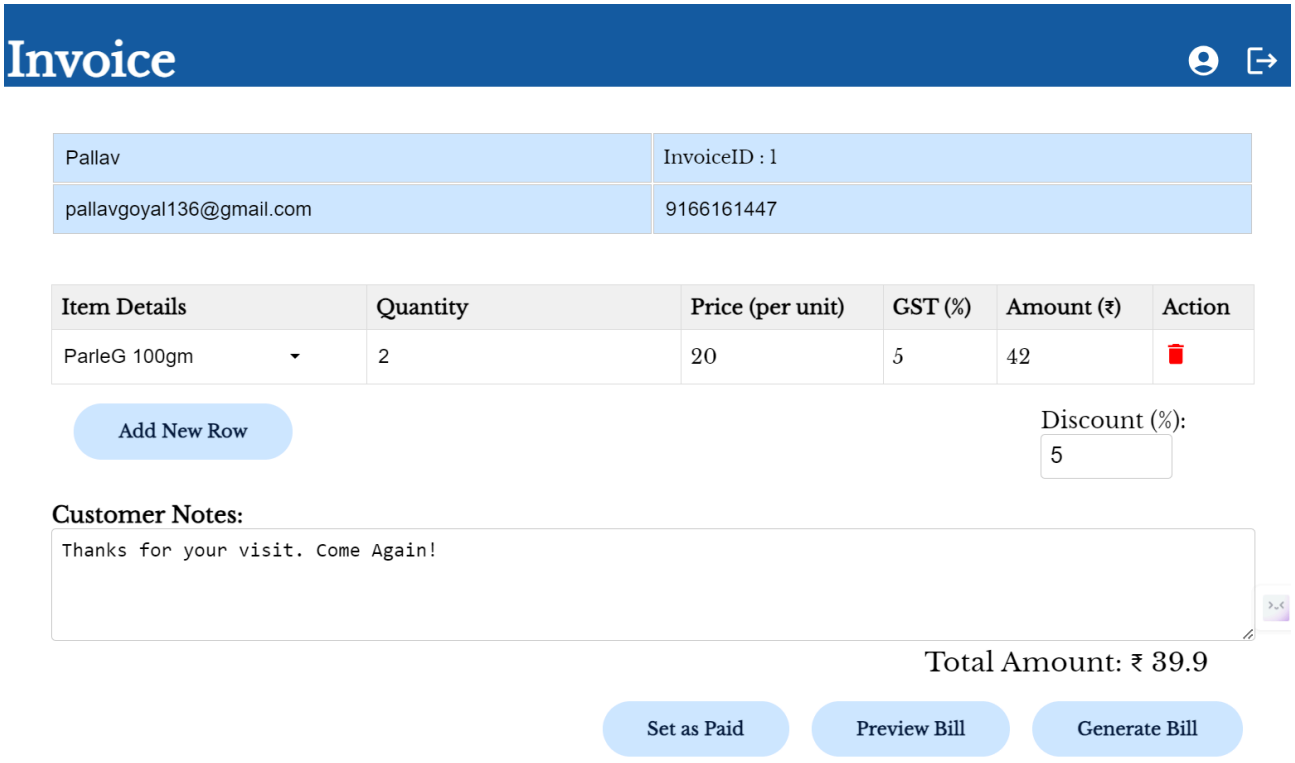
Dashboard updated after generate bill



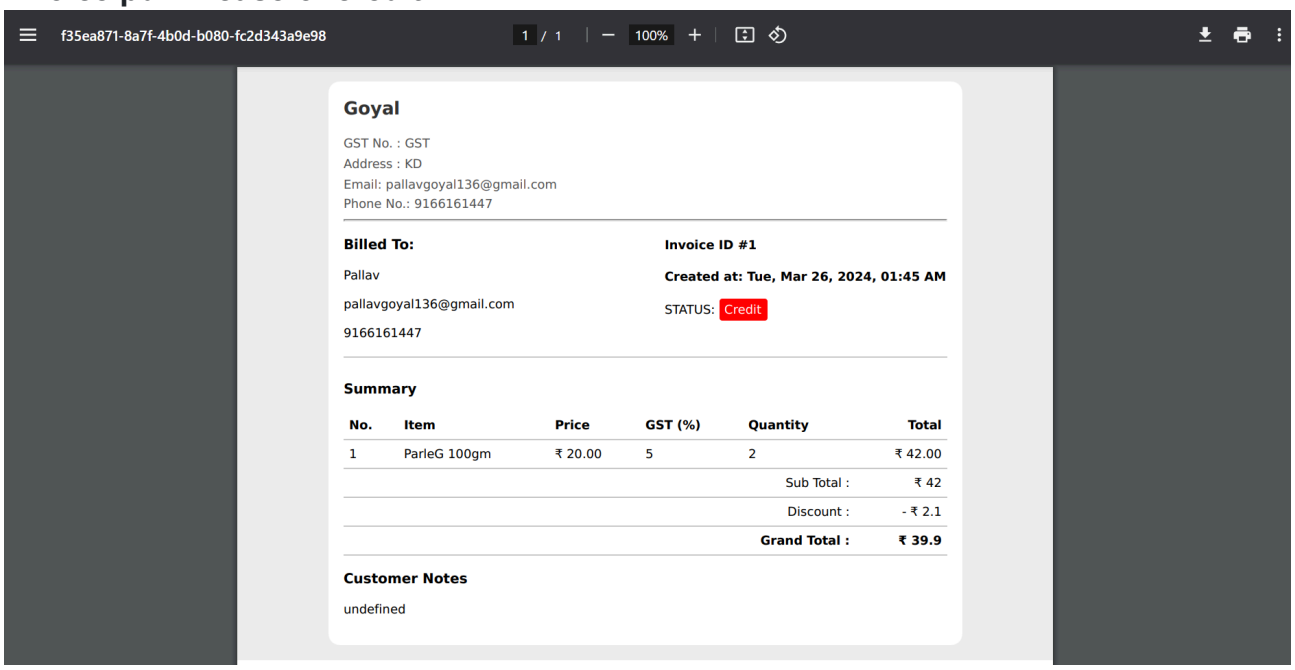
Test #7

Details Items bought through credit. Updated in the pending transactions. Corresponding invoice is also generated and mailed to the customer.

Addition of items and setting it as credit



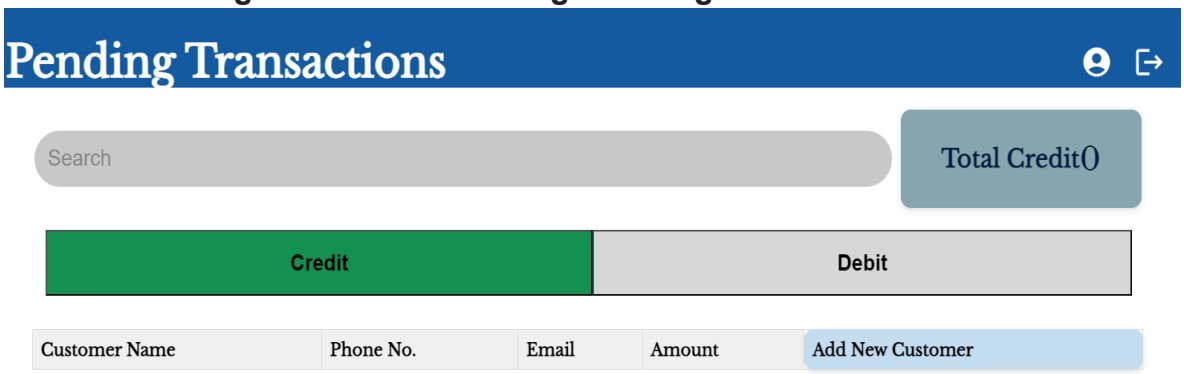
Invoice pdf in case of credit



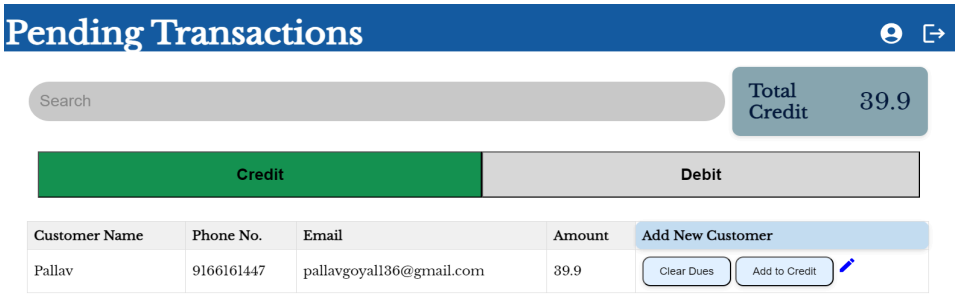
Invoice of existing customer gets added to existing customer database instead of creating a new customer and subsequently adds credit amount to customer

```
_id: ObjectId('6601d71ebff0240b92f7c0bc')
userID: "65f45c105fcad1b212b4bbe3"
name: "Pallav"
phoneNo: "9166161447"
email: "pallavgoyal136@gmail.com"
creditAmount: 39.9
▼ invoiceList: Array (2)
    0: ObjectId('6601d71dbff0240b92f7c0b4')
    1: ObjectId('6601dbdcbff0240b92f7c132')
__v: 1
```

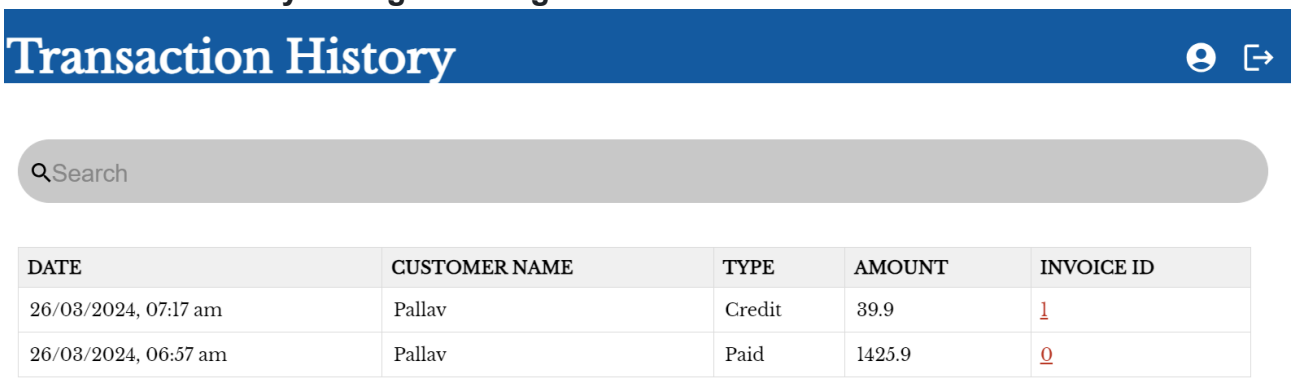

State of Pending transaction before generating a credit bill



Pending Transaction after adding bill on credit



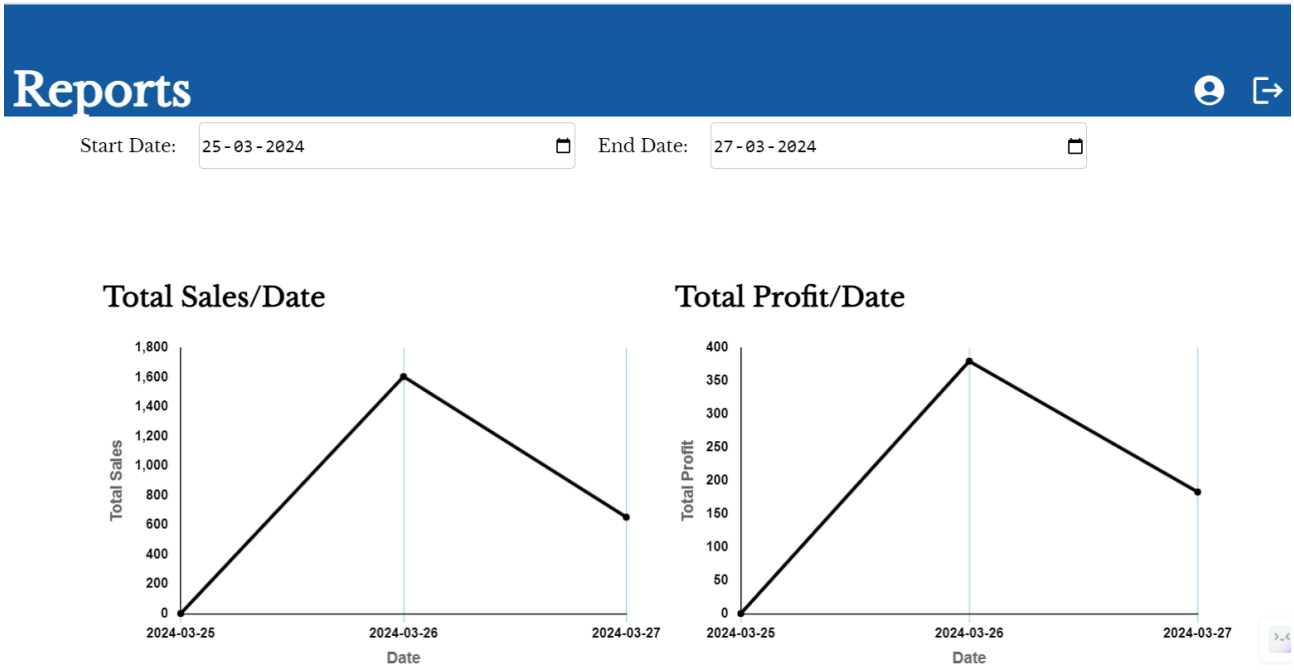
Transaction History after generating credit bill



4. Reports

Module Details: This model ensures smooth functionality of the report generation system as soon as the user chooses the desired range of dates
Test Owner: Pallav Goyal
Test Date: 27/03/2024
Test Results: In this test we validated the successful integration of backend and frontend components of reports which plots graphs on the basis of data received from backend.

Test #1
Details :The dates were added and consequently the reports were shown


Reports
Start Date: 25-03-2024
End Date: 27-03-2024
Total Sales/Date
1,800
1,600
1,400
1,200
1,000
800
600
400
200
0
Total Sales
2024-03-25
2024-03-26
2024-03-27
Date
Total Profit/Date
400
350
300
250
200
150
100
50
0
Total Profit
2024-03-25
2024-03-26
2024-03-27
Date

4. System Testing

FUNCTIONAL REQUIREMENTS

1. Requirement: The user should be able to sign in through his email ID (or phone number) and password if he is already a member.

This sign-in functionality underwent thorough testing to ensure its reliability and effectiveness. We created test cases covering various scenarios, including valid and invalid inputs. These test cases were executed by manually navigating to the sign-in page, inputting different combinations of email IDs and passwords, and observing the system's responses. For instance, we tested the successful sign-in process by entering valid credentials and verifying that the user was redirected to the Dashboard page. Additionally, we intentionally input invalid email formats, non-registered user details, and incorrect passwords to validate, then appropriate error messages(“wrong credentials”) were displayed in each scenario.Throughout the testing process, no bugs or issues were encountered, indicating that the sign-in functionality operates as intended and meets the specified requirements.

Test Owner: Kundan Kumar

Test Date: 26/03/2024

Test Results: The Sign-in page worked as expected.We had tested it on our deployed website. No bug was reported.

Additional Comments: NA

2. Requirement: A new user should be able to sign up by creating his account through filling required details(which include his name, firm's name, phone number, email ID, registered GST number) and setting a valid password.

The sign-up functionality enables new users to create accounts by providing required details, including their name, firm's name, phone number, email ID, and registered GST number, along with setting a valid password. To ensure the effectiveness of this functionality, we devised a series of test cases covering various scenarios. These tests were executed by manually navigating to the sign-up page, inputting valid and invalid data into the respective fields, and observing the system's responses. We verified that users could successfully sign up by providing all required details and setting a valid password, leading to the creation of their account. Additionally, we tested scenarios such as entering incomplete or incorrect information, duplicate email IDs, and invalid passwords(less than required min. length) to ensure proper validation and error handling. One bug was encountered, where the user was able to enter any length of phone number. Throughout the testing process, no other major issues or bugs were encountered, confirming that the sign-up functionality functions as intended and meets the specified requirements.

Test Owner: Kundan Kumar

Test Date: 26/03/2024

Test Results: The Register page worked as expected. One small bug was encountered where any length of phone number was acceptable. We fixed it and made it accept only phone numbers of length 10.

Additional Comments: NA

3. Requirement: There should be an option of ‘Forgot Password?’ in case the user forgets the password. Authentication shall be carried out through OTP verification on the email the user had provided earlier.

The "Forgot Password" functionality provides users with an option to reset their password in case they forget it. Authentication for password reset is carried out through OTP (One-Time Password) verification sent to the email address the user had provided earlier during registration. To ensure the effectiveness of this functionality, a series of test cases covering various scenarios were devised and executed. These tests involved manually navigating to the "Forgot Password" page, inputting the registered email ID, and initiating the password reset process. Upon entering a valid email ID, an OTP should be generated and sent to the user's email address for verification. Subsequently, the user should be prompted to enter the OTP received via email. The system should then validate the OTP entered by the user and allow them to reset their password if the OTP is correct and then we tried to login with a new password , everything worked as expected. Additionally, tests were conducted to verify error handling in scenarios such as entering an invalid or non-registered email ID or wrong OTP , each time it worked as expected. Throughout the testing process, no issues or bugs were encountered, confirming that the "Forgot Password" functionality operates as intended and provides users with a secure and reliable password recovery option.

Test Owner: Kundan Kumar

Test Date: 26/03/2024

Test Results: The Forgot Password page worked as expected.We had tested it on our deployed website. No bug was reported.

Additional Comments: NA

4. Requirement: The dashboard shall display the firm name and the GST number.

When a user is logged in , the Dashboard displays Users Firm Name and GST number. The test cases include registering new users and checking if it is correctly displayed on Dashboard. Everything works as expected no bugs were found.

Test Owner: Kundan Kumar

Test Date: 26/03/2024

Test Results: No bugs were found. Everything works as expected.

Additional Comments: NA

5. Requirement: The dashboard shall provide the user with various options like invoices, inventory, pending transactions, transaction history, FAQs, and contact us.

To check this requirement , after logging in we manually navigated to each of the options making sure each of the options is easily accessible. Everything worked fine , no bugs were detected.

Test Owner: Kundan Kumar

Test Date: 26/03/24

Test Results: The Dashboard provides options to navigate to different pages like invoices, inventory, pending transactions etc. everything works fine and no bugs were detected.

Additional Comments: NA

6. Requirement: The dashboard shall provide the user with the details of his daily sales, daily profit, number bills generated in a day (customer count), and the previous day's sales.

For testing the functional requirement of the dashboard providing details such as daily sales, daily profit, bills count, and previous day's sales, a comprehensive manual testing was done. This included verifying that the dashboard accurately displayed these metrics by comparing them with the expected values derived from test data. Tests were conducted to ensure that daily sales and profit calculations were correct, customer count was accurately recorded, and previous day's sales were retrieved and displayed correctly. By executing these tests, the functionality was thoroughly evaluated, and no discrepancies or bugs were identified, ensuring the reliability and accuracy of the dashboard's reporting capabilities.

Test Owner: Kundan Kumar

Test Date: 26/03/24

Test Results: The Dashboard page correctly displays all the metrics and correctly fetches the previous day records. Everything works as expected no major bugs were found.

Additional Comments: NA

7. Requirement: The dashboard should show the graphical plot of total profits earned per day over a week and total sales per day over a week.

To ensure the functionality of the dashboard in displaying graphical plots of profits earned per day over a week and total sales per day over a week,a lot of different tests were manually performed. The testing process began with data verification to ensure the accuracy of the dataset used for plotting. Subsequently, the dashboard's ability to render graphical plots accurately and intuitively was seen, ensuring that the plots provided a clear representation of the profit and sales trends over the specified week. Data consistency tests were conducted to verify that each data point on the graphical plots correspond correctly to the underlying profit earned or total sales value for each day. We also checked hover-over functionality for detailed information. Through all testing, no issues or inconsistencies were identified, confirming the reliability and accuracy of the dashboard's graphical plotting feature.

Test Owner: Kundan Kumar

Test Date: 26/03/2024

Test Results: The graphical plotting feature on the dashboard accurately presents profits earned and total sales per day over a week, with intuitive and visually appealing plots. No major bugs were found.

Additional Comments: NA

8. Requirement: The dashboard shall also have the Profile option (where the user can change or update his other details except email) and the log-out option.

The Profile and Log-out options on the dashboard underwent different test to validate their functionality. Manual testing confirmed that the Profile option enables users to efficiently update their details, excluding email, with changes accurately reflected in the system. Additionally, the Log-out option securely logs users out of their accounts, terminating the current session and redirects them to the login page for re-authentication. No bugs were found.

Test Owner: Kundan Kumar

Test Date: 26/03/24

Test Results: The profile and log-out option works as expected. No bugs were found.

Additional Comments: NA

9. Requirement: The system shall provide a unique invoice ID for each invoice. This would be the total number of invoices he has created till now (indexed from 0).

This test includes the backend function to get the count of invoices the user has created till that time. Since invoice ID must be unique and auto-generated by the system, the invoice ID is set to the count of invoices of that user. We acted as a general user, on clicking the invoice page, the invoice ID was displayed automatically and correctly. This got updated only on generating a new invoice, which is exactly how it should work.

Test Owner: Pragati Agrawal

Test Date: 27/03/2024

Test Results: The invoice ID gets updated automatically after creating each invoice. We tested it on our deployed website and no bug was reported.

Additional Comments: NA

10. Requirement: The system shall automatically add a new customer to the list of customers, in case a new customer comes for the invoice (i.e. if a new customer email id). If any existing customer comes again, that invoice will be added to the same customer's invoices.

This test includes both the frontend and backend to find or add a customer to the existing list of any user. We acted as a general user and tried both these. First we added a new invoice for a new customer email. This added a new customer in the database. Then we created another invoice for the same customer. This added the invoice to the list of invoices of that customer and changes in amount were reflected in the pending transactions page, if that invoice was given as credit.

Test Owner: Pragati Agrawal

Test Date: 27/03/2024

Test Results: For every new customer (new email), a new customer was added, and for a pre existing customer, the invoice got added to the pre existing list of invoices. This was exactly how it should work. We tested it on our deployed website and no bug was reported.

Additional Comments: Future improvement: To fetch the details of any customer using the customer email, if customer already exists.

11. Requirement: The system shall allow the user to customize an invoice with details like the name of the customer and customer notes. He shall also be able to add items and specify quantity (using a search with the name of the item), and apply discounts(if any).

This test includes testing frontend input boxes, so that they work correctly and take in valid and correct inputs. We acted as a general user and tried to customize a new invoice. We could fill in all the customer details (name, email and phone number) and then select items from the list of available items in the inventory. On entering the quantity of the selected item, the amount for that item and the total amount of the invoice got updated automatically and correctly. The discount entered also correctly updated the total amount of the invoice.

Test Owner: Pragati Agrawal

Test Date: 26/03/2024

Test Results: The invoice page worked as expected. We tested it on our deployed website and almost no bug was reported.

Additional Comments: NA

12. Requirement: The system shall automatically fill the sale price and applicable GST(%) for any item after selecting it from the list of items in the inventory.

This test includes fetching the details of that item from the inventory and filling in the table in the invoice. Once an item is added to an invoice, it should not be added again to the same invoice. Similarly, if an item is deleted from the invoice, it should again be available to be added to the invoice. We acted as a general user, and test these. Everting worked seamlessly.

Test Owner: Pragati Agrawal

Test Date: 27/03/2024

Test Results: The details were filled in correctly. We tested it on our deployed website and no bug was reported.

Additional Comments: NA

13. Requirement: If the selected quantity of an item is greater than the quantity available in inventory(including all batches), the system shall generate an error message saying that the required quantity is greater than that available in inventory.

This test includes both the frontend and backend, to check if the quantity of any item entered in the invoice is feasible. We tried to add quantity of an item more than its available quantity, and the website displayed an alert message, saying that “Quantity exceeds the available stock”.

This was tested in integration testing “Invoice, Test#1”. This quantity includes the total non-expired quantity of each batch in the inventory.

Test Owner: Pragati Agrawal

Test Date: 26/03/2024

Test Results: The website shows an error if the quantity exceeds the available quantity in the inventory. We tested it on our deployed website and no bug was reported.

Additional Comments: The quantity of any item in the invoice gets reduced from the batch having the earliest expiry date.

14. Requirement: The software shall automatically calculate the total amount based on the quantity of each item and after applying tax and discount (if any).

This test includes the frontend component of the total amount in the invoice. This value needs to be updated correctly on changing the quantity of any pre-existing item in the invoice, addition/deletion of any item from the invoice, changing any item in a pre-existing row or changing the value of the overall discount. We acted as a general user and tried all these modifications in the invoice, and in each case, the total amount got updated correctly.

Test Owner: Pragati Agrawal

Test Date: 26/03/2024

Test Results: We tested the total amount component in many ways on the deployed website and it worked perfectly fine in all cases.

Additional Comments: Initially some bugs were there in the total amount calculation. We fixed those bugs through recursive programming, rounded off each value to 2 decimal places, and asked another team of 2-3 members to check it. All bugs found have been resolved.

15. Requirement: The system should provide an option to add as credit in case the customer wants to pay later. This option shall automatically update the credit log in the pending transactions page of the user.

This test includes the frontend and backend functions. On adding an invoice as credit, the system should mark it as “credit” and should add this amount to the prevailing total dues of that customer. This invoice should also be marked as “credit” on the transaction history page. We acted as a general user and tried to add an invoice as credit for a pre-existing customer, and it worked perfectly. If it is the first credit invoice of any customer, a new entry will be created in the pending transactions page, otherwise the amount will simply be added to the total pending amount of that customer.

Test Owner: Pragati Agrawal

Test Date: 26/03/2024

Test Results: We tested the “Add as credit” feature of invoice on our deployed website, and it worked perfectly fine. No bug was reported.

Additional Comments: NA

16. Requirement: The software shall allow the user to preview and generate the bill. With the generate bill button, the system shall add a new invoice to the database, send the invoice via email to the customer, make a new entry in the transaction history page, make required changes in the inventory and add to pending transactions, if necessary.

This test includes both frontend and backend functions to work. We acted as a general user and tested the preview bill button. The system showed us a preview of the invoice we had created (All invoice fields must be filled). Similarly, the generate bill button added a new invoice to the database of that user, sent a mail to the customer (on the email ID given in the invoice), added this invoice to the Transactions History page and reduced the number of items in the inventory. If it was a credit invoice, this was also updated on the Pending Transactions page.
All these tests are also shown in Integration Testing.

Test Owner: Pragati Agrawal

Test Date: 26/03/2024

Test Results: We tested the preview bill and generated bill functionalities on our deployed website and no bug was reported. All required changes were made seamlessly and smoothly.

Additional Comments: The pdf generation feature had some bugs initially, and pdf was not being displayed on the deployed website. Then we tried a variation of our original method and it worked. We had also improved the look of the invoice from the initial version.

17. Requirement: The Inventory page should include a list of items, a search bar, a filter button, and an "Add New Item" option. For a particular item the following columns should be shown: Item Name/ID, Sale Price, Purchase Price, Stock.

The test checked for all the mentioned components in the deployed website. We acted as a general user and tried to access the inventory page. All the items of the user are fetched from the database and shown as a list. All the details - Item ID, Item Name, Sale Price, Cost Price, Stock are correctly shown in the table under appropriate columns. An "Add New Item" button was present which can be used to add items in the inventory. Filter buttons were present alongside each column heading to sort the items according to that column and a Search bar was also present at the top.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: The invoice page contains all the components required in the SRS. All the data shown in the inventory table is correctly fetched from the database.

Additional Comments: NA

18. Requirement: The system should allow the user to click on the search bar and search items by name/ID. Also, it should allow the user to click the filter option and apply filter(s) on the results of the search bar

Various inputs in the search bar were entered, various filters were applied, and test cases consisting of a combination of these were also tested on the deployed website.
Upon searching in the search bar, only the items with Item Name containing the input string were shown in the list. When filters based on Item ID, Item Name, Sale Price, Cost Price, Stock were applied, the items could be arranged in ascending or descending order of the filter selected. If filter was applied after an input string was entered in the seach bar, the filter worked correctly on the search results.
Also the items were colour-coded and sorted based on their expiry date as a default setting.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: Search bar and filters worked fine. Filters worked even on the search results of the search bar. Items were sorted based on expiry date by default. Other searches could be applied. Almost no bugs were reported.

Additional Comments: NA

19. Requirement: The system should allow the user to click on the “Add New Item” option to add new items to the inventory. Adding a new item should require filling in the following fields: Item Name/ ID, Quantity, Sales Price per unit, GST, Category, Cost Price per unit, and Batch Expiry.

We acted as a general user and tried to add new items to the inventory on the deployed website. On clicking the “Add New Item” button, a dialog box opens where all the details can be filled. If all the required fields are not added, appropriate error messages are displayed. On filling all the details - Item ID, Item Name, Sale Price, Cost Price, GST and category, and clicking on “Save”, a new item with zero stock is added in the list. Batches for an item were added using the “Add Batch” button in the list. On clicking it, a dialog box

appeared. After entering the batch ID, quantity and batch expiry and clicking on “Save”, the item was automatically colour-coded based on the latest batch expiry. Also the total stock of the item correctly displayed the sum of quantity in all the batches.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: Adding item in inventory feature using “Add New Item” button and adding batches work as expected. Almost no bugs were reported.

Additional Comments: NA

20. Requirement: The system should allow the user to view the item profile, which should include all the above item details, batches, and their expiry. Options to edit or delete the item should also be available.

There was a column in the table of items, containing four buttons - “Edit Item”, “Add Batch”, “View Batch”, “Delete Item”. All these buttons were tried. Upon clicking “Edit Item”, a dialog box similar to “Add New Item” was opened in which changing the details was possible. All the changes were reflected after clicking on “Save”.
Upon clicking “View Batch”, a dialog box appeared which displayed all the batches of the item (and their details). Each batch had an “Edit Batch” option which opened a dialog box similar to “Add Batch”. On entering all the details and clicking on “Save”, batch was updated. Each batch also had a “Delete Batch” option which could be used to remove a particular batch.
Upon clicking “Delete Item”, the item was removed from the database as well as the list of items.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: Options for viewing, editing and deleting items and their batches are available as required by the SRS. All features work as expected. Appropriate error messages are displayed if required fields are not entered. All the changes are also correctly reflected in the database.

Additional Comments: “Add Batch” button was tested along with “Add New Item” button, hence it is not mentioned again here.

21. Requirement: The Pending transactions page shall show the list of customers with pending credit along with search and filter options. For a particular customer, the list shall show the name/ID, phone number, email and total credit.

The test checked for all the mentioned components in the deployed website. We acted as a general user and tried to access the Credit tab on the Pending Transactions page. All the customers of the user with pending credit are fetched from the database and shown as a list. All the details - Customer's name, email, phone number and total credit are correctly shown in the table under appropriate columns. Filter buttons were present alongside each column heading to sort the items according to that column and a Search bar was also present at the top.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: The Pending Transactions page contained all the components as expected. All the data was correctly fetched and displayed from the database.

Additional Comments: Since each user is identified by a unique email, an explicit customer ID was redundant and hence removed.

22. Requirement: The system shall allow the user to search for a particular customer by Name/ID/ Phone Number,/email. He can use the filter option to narrow down the results of the search option.

Various inputs in the search bar were entered, various filters were applied, and test cases consisting of a combination of these were also tested on the deployed website.
Upon searching in the search bar, only the customers with Customer Name containing the input string were shown in the list. When filters based on email, phone number and credit amount were applied, the customers could be arranged in ascending or descending order of the filter selected. If filter was applied after an input string was entered in the search bar, the filter worked correctly on the search results.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: Search bar and filters worked fine. Filters worked even on the search results of the search bar. Almost no bugs were reported.

Additional Comments: NA

23. Requirement: The user shall be able to use the "Add New Customer" option to add a new customer's record manually. While adding a new customer, the following details are to be filled: Name, Phone Number, email and amount. A notification containing these details shall be sent to the customer for authentication.

There is an “Add New Customer” button, which can be used to manually add a new customer. On clicking it a dialog box opens, in which all the fields can be entered. On clicking “Save”, a new entry in the table (and in the database) is created. An invoice showing the credit amount is created and added to the database (this can also be viewed on the Transaction History page along with other invoices). An email with a pdf of this invoice as attachment is automatically sent to the Customer’s email.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: “Add New Customer” button works as expected and fulfills the requirements mentioned in the SRS. Appropriate error messages are displayed if required fields are not entered or if the email entered is invalid.

Additional Comments: NA

24. Requirement: The user shall be able to edit the credit log and details of the customer.

There are options - “Clear Dues”, “Add to Credit” and “Edit Customer”, which allow the user to clear credit, add credit and edit customer details (email and phone number) respectively. On clicking “Clear Dues” or “Add to Credit” button, a dialog box opens, where user can enter the amount and click on “Submit” to reflect the changes in the database. Appropriate error message is shown if amount to be cleared is more than pending credit amount. On clicking the “Edit Customer” option, a dialog box appears allowing the user to change email and phone number of the customer. Changes are reflected are clicking on Submit.
Any user with zero credit is automatically removed from the credit list. Also any change made in the credit amount (“Clear Dues” or “Add to Credit”) is recorded as an invoice in the database (this can also be viewed on the Transaction History page along with other invoices). An email with a pdf of this invoice as attachment is automatically sent to the Customer’s email.

Test Owner: Dhruv Gupta

Test Date: 26/03/2024

Test Results: All the functionalities are working as expected. No bugs were reported.

Additional Comments: NA

25. Requirement: The Pending transactions page shall show the list of suppliers with pending debit along with search and filter options. For a particular supplier, the list shall

show the name/ID, phone number, email and total debit. The user shall be able to - search for a supplier using the search bar, apply filters on the search results, add a new supplier, view/edit the debit log and details of any supplier.

All these components and functionalities could be accessed on Debit tab of the Pending Transactions page. All the functionalities are similar to the Credit tab functionalities (except that no email is sent to the supplier in case of editing the debit amount). Search and filters worked as expected. The options “Add Supplier”, “Remove Amount”, “Add Amount” and “Edit Supplier” work in the same way as the options “Add New Customer”, “Clear Dues”, “Add to Credit” and “Edit Customer” on the Credit tab respectively. Every change in amount is also recorded as an invoice in the database and can be viewed on the Transaction History page.

Test Owner: Dhruv Gupta

Test Date: 27/03/2024

Test Results: All the functionalities are working as expected. No bugs were reported.

Additional Comments: All the functionalities were tested in the same way as the Credit tab functionalities. Hence detailed test results are not mentioned again.

26. Requirement: The system shall show the list of all invoices created by the user on the Transactions History page. This list will also include the invoices generated due to modifications in the amount of any customer on the Pending Transactions page. These entries in the table can also be filtered on the basis of customer name.

This test includes the backend function to get all the types of invoices generated: “Paid”, “Credit”, “Credit dues cleared” and “Amount added to credit”. The first two modes come from the two payment modes on the Invoice page. The remaining two modes come from the two features in the Pending Transactions, if some amount is deducted from the remaining credit of that customer, then “Credit dues cleared” otherwise if it is added then “Amount added to credit”. We acted as a general user and created all these types of transactions.

All the invoices were correctly added to this page. The invoice ID was also incremented correctly. A search bar is present to filter the entries in the transactions history page, on the basis of customer name.

Test Owner: Pragati Agrawal

Test Date: 27/03/2024

Test Results: We tested it on our deployed website and all kinds of invoices were shown as expected. The search bar also worked perfectly.

Additional Comments: The time displayed on the invoice generation through pending transactions page had a bug. It has been resolved.

27. Requirement: The system shall show enable the user to view the PDF of any invoice (Paid, Credit, Credit dues cleared, Amount added to credit) by clicking on the link in the invoice ID column in each invoice on the Transactions History page.

This test includes backend to fetch the invoice details to be displayed using the invoice ID, and to create a PDF of it to display on the website. We acted as a general user and tried to open all these kinds of PDFs of invoices, and everything works smoothly.

Test Owner: Pragati Agrawal

Test Date: 27/03/2024

Test Results: We tested it on our deployed website and all kinds of invoices were viewed as PDF correctly. All details were rendered correctly. Almost no bug was reported.

Additional Comments: The time displayed on the invoice generation through pending transactions page had a bug. It has been resolved.

28. Requirement: The software system shall provide users with the capability to access comprehensive reports detailing Total sales, Total profit, and Number of bills for each date within specified date ranges. Furthermore, users shall have the ability to view the distribution of bills categorized as either paid or credit.

To check the functionality outlined in the requirement, a lot of different manual tests were performed. This involved a series of tests aimed at ensuring accurate reporting of Total sales, Total profit, Number of bills, and bill distribution within specified date ranges by manually changing system date and generating invoices on different dates and data shown by reports were manually verified. No bugs were found, everything works as expected.

Test Owner: Kundan Kumar

Test Date: 27/03/2024

Test Results: The Reports section was tested on our deployed website and no bugs were found. All the details shown were coherent with the data.

Additional Comments: NA

29. Requirement: The FAQs page must display a list of Frequently Asked Questions and their answers, helping the user with common queries related to the software.

On opening the FAQs page using the Navbar, a list of questions and their answers is displayed. All the questions are valid and answered satisfactorily.

Test Owner: Dhruv Gupta

Test Date: 27/03/2024

Test Results: All the components were correctly rendered and displayed on the deployed website. No bugs were reported.

Additional Comments: NA

30. Requirement: In case the user faces any issue, he should be able to contact the developing team using this feature and if the user wants any information about the software/development team, he should be able to use this feature to reach out.

The "Contact Us" feature facilitates seamless communication between users and the development team, allowing users to address any issues, queries, or requests for information about the software or development team. By filling out a contact form that includes fields for their name, email, and message, users can submit their queries directly to the team. Thorough testing ensures that users can successfully submit queries and the development team receives queries through mail, few bugs were encountered , where users can send queries to the team without filling up their name details and email. These bugs were fixed making these fields mandatory and adding an additional check for valid email syntactically. Additionally, the contact information section prominently displays the team's phone number for easy access. Upon successful submission of a query, users receive a confirmation message, enhancing their confidence in the effectiveness of the communication channel. Overall, the "Contact Us" feature enhances user experience by providing a reliable and efficient means of reaching out to the development team for assistance or information.

Test Owner: Kundan Kumar

Test Date: 26/03/24

Test Results: The Contact Us page worked as expected.Few minor bugs were detected , where users can get away with sending queries to the development team without providing their Name and contact Email.These fields are now made mandatory.

Additional Comments: NA

NON FUNCTIONAL REQUIREMENTS

This includes requirements related to how the software performs under different load conditions, response times, maintainability, scalability etc.

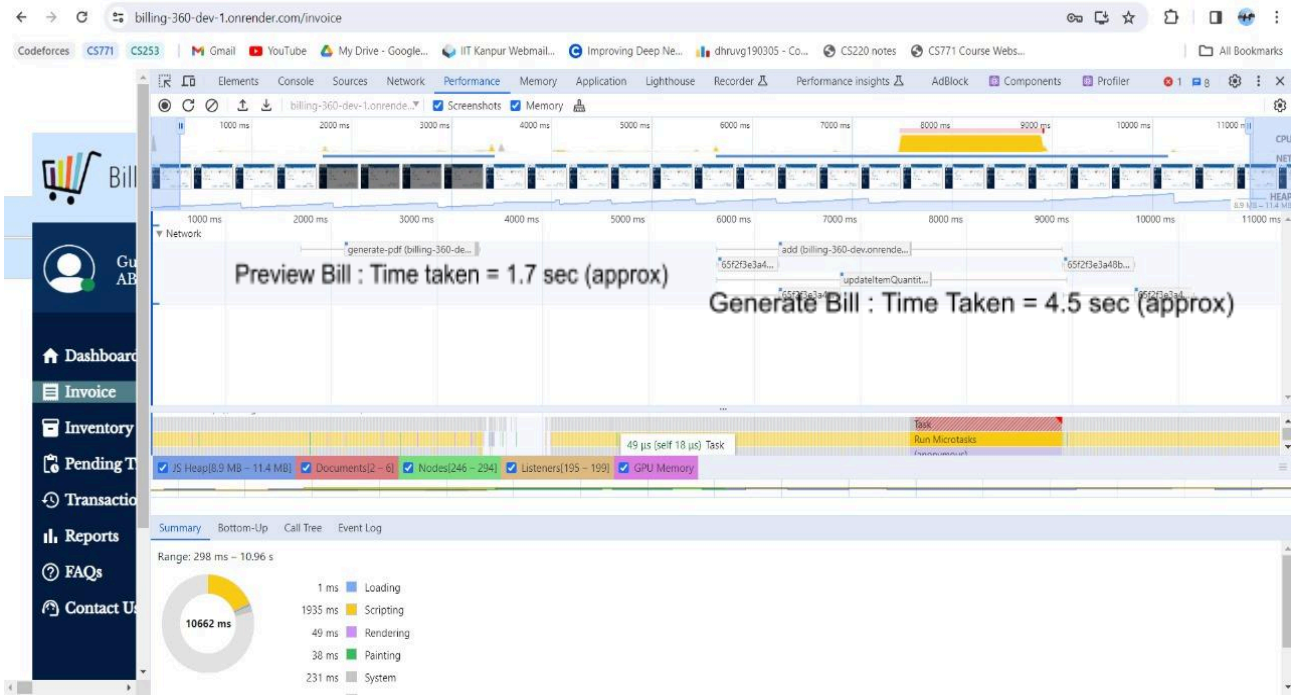
Test Owners: Pragati Agrawal, Dhruv Gupta and Kundan Kumar

Test Dates: 24/03/2024 - 28/03/2024

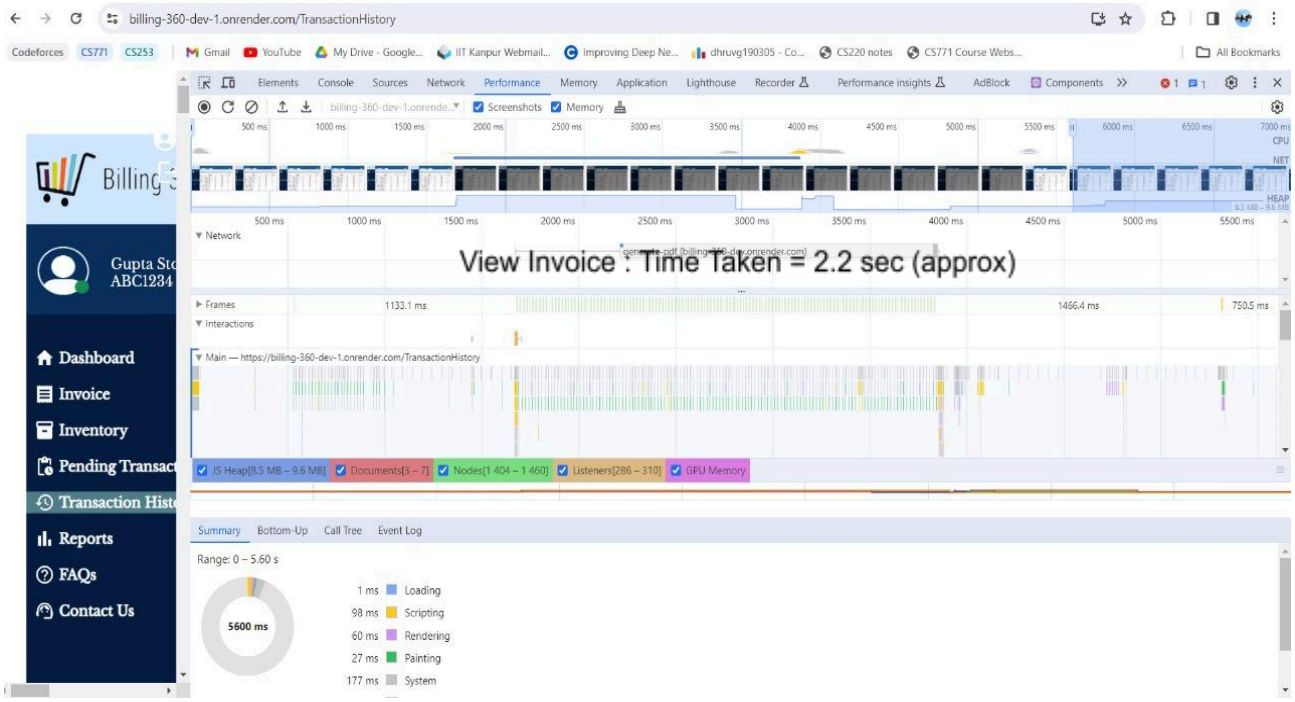
Test Results:

1. Performance Requirements:

- *The search operation used while adding items to a bill should not take more than 0.5 seconds*
  The search operation on all search bars works almost instantaneously without any time delay.
- *The software should allow no less than 10000 entries in the inventory and no less than 10000 entries in the credit and debit log each*
  No limit is applied on the number of entries in the database as of now. However total space occupied by all the users combined is capped by 512 MB (due to limited cloud storage available).
- *History until at least 3 years ago should be available for generated bills and reports*
  Automatic data cleaning is not implemented, hence if any cleanup required has to be done manually. Hence no data is removed by the system itself no matter when the data was created.
- *Preview bill and Generate bill functionality (i.e., the creation of pdf and showing it on web) should not take more than 6 seconds, provided the user has a stable internet connection.*
  Time was noted for pdf creation using Chrome DevTools on various pages - Invoice and Transaction History. Following images show screenshots for one of the tests. Maximum time over multiple test cases never exceeded 5.5 sec for Generate bill, and 3.5 seconds for Preview bill and View Invoice.



Time taken by Preview bill and Generate bill on Invoice page.



Time taken by View Invoice (clicking on invoice ID) on Transaction History page.

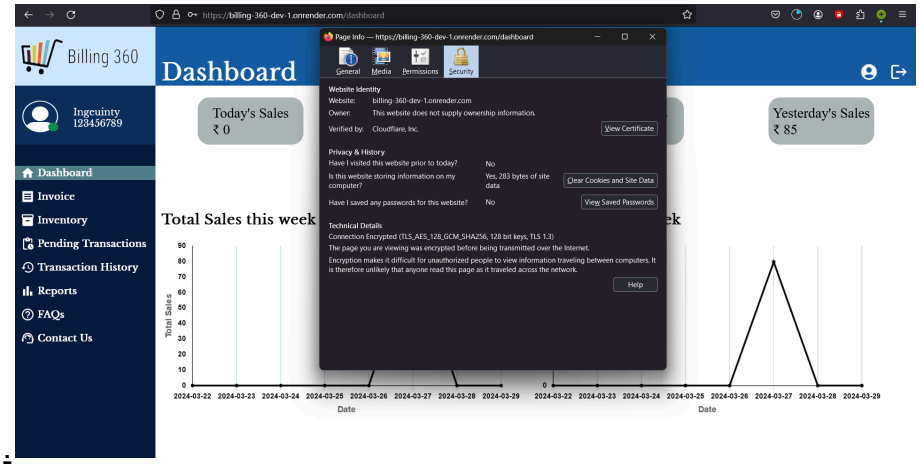
2. Safety and Security Requirements:

- The software authenticates the user during login using a strong password. A strong password would require at least 8 characters.
  In the Sign-up page we put a restriction of at least 8 char on the password field, without fulfilling this requirement signup is not allowed.
- There is a provision to set a new password in case the user forgets the password. In such a case, authentication is via OTP (One Time Password) sent to the user's registered email. The OTP shall be valid for 2 minutes. A similar procedure is followed for password reset.
  To validate the OTP-based password reset functionality as per the specified requirement, we tested it manually. The system's ability to generate unique OTPs and deliver them to users' registered email addresses was confirmed after testing, along with verification of the OTP validity period set at 2 minutes. The complete password reset process, from user-initiated requests to OTP-based verification, was tested, covering different edge cases such as different email is provided which is not registered , wrong OTP entered , OTP entered after 2 minutes.Each time our deployed website behaved as expected, maintaining the security. Additionally, security measures implemented for OTP generation and transmission were evaluated like OTP should be very random each time. Throughout testing, no bugs or issues were encountered, affirming the reliability and effectiveness of the OTP-based password reset feature, which provided users with a secure and seamless password recovery experience.
- Passwords are stored securely by hashing so that even the administration can only verify if the password is correct without discovering the password.


```
_id: ObjectId('6603f24e9e6b1e0a4d4f389b')
firstname: "Doraemon"
lastname: "Gupta"
email: "kundank22@iitk.ac.in"
password: "$2b$10$fe6hrH15leN8mk7oiQLJBO62DSpMtM8dPjUz1uzqUmyeueQIenqgC"
gstno: "ABGH"
shopname: "Gian Shop"
shopaddress: "School ki pechhe wali pahar"
phonenumber: "1234567890"
__v: 0
```


  As can be seen in the database, password is stored in a secure manner by hashing. Even the admin doesn't know the user password. Keeping the user's privacy maintained.

- All connections to the server should use Transport Layer Security (TLS 1.2/1.3) encryption.All transaction data should be encrypted properly.



- This is from our deployed website. As can be seen from the screenshot , the site uses TLS 1.3.Therefore all the communication happens between server and client securely.

3. Maintainability:

The software has been designed in an organized manner so that new features can be added and modifications can be done very easily within two working days. Various components of the software are well organised into folders and comments are added wherever required streamline future improvements.

4. Reliability:

The system operates with remarkable robustness, consistently delivering reliable performance. The anticipated downtime of the website is significantly minimized, ensuring uninterrupted accessibility for users. This reliability enhances the overall user experience and fosters trust in the website's functionality.

5. Scalability:

We've tested the system with 15-20 concurrent users and it performed well. However, to accurately gauge its scalability, it needs to be made available to the general public. Only through widespread usage can we gather sufficient data to assess its performance under higher loads.

6. Portability:

Being a web-based app, the software is easily accessible from any device with browser support. We tested the deployed website, and it works perfectly fine on Google Chrome, Mozilla Firefox and Microsoft Edge web browsers.

5. Conclusions

- **How Effective and exhaustive was the testing?**

  Testing was done almost exhaustively. Each API underwent testing to ensure that they were modifying the database and returning the desired information when needed. We have taken care of all cases, thus ensuring complete branch coverage. The frontend was also tested and bugs were resolved. Validity of each input has been checked in all forms. Appropriate alerts have been shown in such cases. Edge cases were also tested and were confirmed to work as expected. For example, when the entered start date is ahead of the entered end date while viewing reports, an alert is displayed preventing the user from dealing with such invalid inputs.

  The testing was quite effective. A developer tested a component that was not developed by him/her. Every component was tested by atleast two developers.

- **Which components have not been tested adequately?**

  Our test coverage for non functional requirements outlined in the SRS document could be further expanded. Perform tests adequately for non-functional requirements like reliability, maintainability and ease of learning could not be performed.

- **What difficulties have you faced during testing?**

  The main difficulty faced during testing was ensuring non-functional requirements are satisfied, in system testing.

- **How could the testing process be improved?**

  The testing could have been improved by doing the unit testing or integration testing as automated rather than manual. The developer might subconsciously assume a few crucial things which may be not so obvious to an end-user or someone who was not involved in the development of the software.

Appendix A – Group Log

Date	Timing	Duration	Agenda
22/03/2024	12:00 - 14:00	2.5 hours	• Discussed various types of testing. • Distributed the tasks of various types of testing among subgroups within the team.
24/03/2024	14:00 - 16:00	2 hours	• Discussed the various doubts faced by each subgroup in their respective tasks.
26/03/2024	11:00 - 13:00	2 hours	• All the group members met in an online meeting. • Every subgroup updated each other with their work.
29/03/2024	15:00 - 16:00	1 hour	• Discussed doubts among team members.
31/03/2024	20:30 - 23:00	2.5 hours	• Reviewed the testing document. • Cleared various doubts in the testing document. • Finalised the testing document.