Unit Testing

Unit testing was carried out using **Jest** to ensure each part of the system works exactly as intended in isolation. The focus was on verifying that individual controllers, middleware, and utility functions behave correctly without depending on real database or network interactions.

Independent Controller Testing:

- Each controller — including Auth, Invoice, Order, Payment, Product, Report, and Settings — was tested separately.
- Database models like User, Order, and Invoice were mocked to simulate both successful and failing scenarios, ensuring that each controller handled responses and errors correctly.

Middleware & Utility Validation:

- Core middleware such as auth.js and utility functions like jwt.js were tested to confirm proper token handling, authentication flow, and response behavior.
- This helped verify that security-related components functioned reliably under various edge cases.

Error Handling & Edge Cases:

- Tests were written for both expected and unexpected outcomes — covering HTTP response codes such as 200, 400, 403, 404, and 500.
- This ensured the system responds gracefully to invalid inputs, missing tokens, or database errors.

Clean Test Environment:

- The testing environment was reset before every test using beforeEach and jest.clearAllMocks(). This guaranteed that each test ran independently with a fresh req and res setup, preventing any side effects from previous executions.

Test file breakdown

1 Auth Controller (authController.test.js)

Focused on user registration and authentication workflows. Mocked User model and bcrypt dependencies were used to validate scenarios like successful signup/login, duplicate email handling, invalid credentials, and database failure cases. Tests ensured proper use of status codes (200, 400, 401, 500) and JWT generation logic.

Registration

Test Case	Expected Status	Notes
Register New User	201	Successfully registers a new user
Missing required fields	400	Required fields are not provided
Email already exists	400	Duplicate email should fail
Handle server errors	500	Server-side errors are handled
Hash password before saving user details	N/A	Ensure password is hashed before storing

Login

Test Case	Expected Status	Notes
User should login with valid credentials	200	Successful login returns user info
Email or password missing	400	Missing credentials should fail
User does not exist	400	Attempt to login with non-existent user
Password incorrect	400	Incorrect password should fail
Return token on successful login	200	JWT token returned on success

Me

Test Case	Expected Status	Notes
Return information about current user	200	Should return authenticated user info

2 Invoice Controller (invoiceController.test.js)

Verified invoice creation, retrieval, and payment status management. Tests simulated multiple invoice states (unpaid, paid, invalid IDs) and confirmed accurate JSON responses and error messages. The controller's interactions with Order and Invoice models were mocked for isolation.

Create For Order

Test Case	Expected Status	Notes
Invoice should create for order successfully	201	Invoice is generated for a valid order
Order not found	404	Attempt to create invoice for non-existent order
User is not order wholesaler	403	Only wholesaler of the order can create invoice

Get Invoice

Test Case	Expected Status	Notes
Get invoice for retailer	200	Retailer can fetch their invoice
Get invoice for wholesaler	200	Wholesaler can fetch invoice
Invoice not found	404	Invoice ID does not exist
User is not related to invoice	403	Users not related to the invoice cannot access it

List for Retailer

Test Case	Expected Status	Notes
Invoices should list for retailers	200	Returns all invoices for retailer
Return empty array if no invoice exists	200	Returns empty list if retailer has no invoices

List for Wholesaler

Test Case	Expected Status	Notes
Invoices should list for wholesaler	200	Returns all invoices for wholesaler
Return empty array if no invoice exists	200	Returns empty list if wholesaler has no invoices

3 Order Controller (orderController.test.js)

Ensured correct order creation, updating, and fetching logic. Tests validated that order totals were computed correctly, associated user roles were respected, and invalid or missing input was handled gracefully. Edge cases like invalid product references and permission denials were also covered.

Order Create

Test Case	Expected Status	Notes
Create an order successfully	201	Order is created with valid data
Wholesaler ID missing	400	Cannot create order without wholesaler ID

Items array is empty	400	Cannot create order without items
Product not found	400	One or more products in order do not exist
Insufficient stock	400	Product stock is insufficient to fulfill order
Handle server errors	500	Server-side errors are handled gracefully

List for User

Test Case	Expected Status	Notes
Order should list for retailer	200	Returns all orders for retailer
Order should list for wholesaler	200	Returns all orders for wholesaler
Return empty array if no orders	200	Returns empty array if user has no orders

Get Order

Test Case	Expected Status	Notes
Get order for retailer	200	Retailer can fetch their order
Get order for wholesaler	200	Wholesaler can fetch their order
Order not found	404	Order ID does not exist
User is not related to order	403	Users not related to the order cannot access it

Update Order Status

Test Case	Expected Status	Notes
Order should update successfully	200	Wholesaler updates order status successfully
Order not found	404	Cannot update non-existent order
User is not order wholesaler	403	Only the order's wholesaler can update its status

4 Payment Controller (paymentController.test.js)

Simulated the complete Razorpay payment workflow — from order creation to signature verification. Environment variables for Razorpay keys were mocked, and tests validated correct handling of paid/unpaid invoices, unauthorized access, and invalid payment signatures.

Razorpay Order Creation:

Test Case	Expected Status	Notes
The invoice does not exist	404	Attempt to create a Razorpay order for a non-existent invoice
The logged-in user does not own the invoice	403	Only the user linked to the invoice can initiate payment
The invoice has already been paid	400	Prevents duplicate payment attempts
The payment gateway credentials are missing or not configured	500	Server error if Razorpay API keys are not set or invalid

Payment Verification

Test Case	Expected Status	Notes
Valid payment signatures	200	Successfully verifies payment and updates invoice status to **"paid"**
Invoice not found for order	404	Invoice corresponding to payment not found
Invalid payment signatures	400	Rejects invalid payment signature during verification

5 Product Controller (productController.test.js)

Checked CRUD operations for products, verifying that product creation, updates, and deletions responded correctly under valid and invalid inputs. Mocked the Product model to confirm filtering, validation, and 404 responses when products were missing.

Product List

Test Case	Expected Status	Notes
List all products for retailer	200	Retailer should be able to view all available products
Filter products for wholesaler to show only their products	200	Wholesaler should only see their own listed products

Filter products by category	200	Returns products that match the selected category
Return empty array if no products found	200	Should return an empty list when no products are available

Product Create

Test Case	Expected Status	Notes
Create a product successfully	201	Product is created successfully with valid input
Set wholesaler automatically from req.user	201	The wholesaler field is automatically assigned from the logged-in user
Handle database errors	400	Returns an error when database operation fails

Product Get

Test Case	Expected Status	Notes
Get a single product successfully	200	Retrieves the details of a valid product
Product not found	404	Returns error if product ID does not exist

Product Update

Test Case	Expected Status	Notes
Update a product successfully	200	Updates product details correctly
Product not found	404	Returns error if product to update does not exist

Product Remove

Test Case	Expected Status	Notes
Delete a product successfully	200	Product is deleted successfully
Return success even if product does not exist	200	Returns success even if product was already deleted or missing

6 Report Controller (reportController.test.js)

Tested report generation logic for different users. Verified proper data aggregation, date-based filtering, and role-based access control. Mocks ensured that no actual database queries were executed while maintaining accurate business logic flow.

Sales Report

Test Case	Expected Status	Notes
Calculate total sales correctly	200	Accurately computes total sales across all orders
Filter sales by wholesaler query parameter	200	Returns sales data only for the specified wholesaler
Return zero sales if no orders	200	Should return a valid response with total sales as 0

Inventory Report

Test Case	Expected Status	Notes
Return all products for wholesaler	200	Lists all products belonging to the wholesaler
Filter products by wholesaler query parameter	200	Filters inventory based on query parameter

Return empty array if no products	200	Returns an empty list when wholesaler has no products

Customer Report

Test Case	Expected Status	Notes
Return unique customers for wholesaler	200	Displays a unique list of customers who have placed orders
Filter customers by wholesaler query parameter	200	Filters customers for the specified wholesaler
Use req.user._id if wholesaler not provided in query	200	Automatically uses the logged-in wholesaler's ID
Return empty array if no customers	200	Returns an empty list when there are no customer records

7 Settings Controller (settingsController.test.js)

Validated settings management, including fetching, updating, and saving user or system configurations. Tests ensured that updates persisted correctly, invalid fields were rejected, and appropriate success or error messages were returned.

Get Profile

Test Case	Expected Status	Notes
Return current user profile	200	Successfully fetches the logged-in user's profile details
Return user profile without password field	200	Ensures sensitive password data is excluded from response

Update Profile

Test Case	Expected Status	Notes
Update user profile successfully	200	Updates user details with valid input
Update only provided fields	200	Partial updates should not overwrite missing fields
Prevent password update through profile update	200	Password field should be ignored if passed in request

Change Password

Test Case	Expected Status	Notes
Change password successfully with valid old password	200	Verifies user identity and updates password correctly
Return error if old password is incorrect	400	Rejects incorrect old password
Return error if oldPassword is missing	400	Validates required field: oldPassword
Return error if newPassword is missing	400	Validates required field: newPassword
Hash new password before saving	200	Ensures password is encrypted before saving to DB

8 Auth Middleware(auth.test.js)

Authentication

Test Case	Expected Status	Notes
Should return 401 if no token is provided	401	Ensures middleware blocks requests missing an Authorization token
Should return 401 if token is invalid	401	Verifies JWT validation logic rejects invalid or tampered tokens

Should return 401 if user is not found	401	Confirms token's user ID is verified against actual database records
Should authenticate valid user and attach to req.user	200	Validates that a valid token decodes correctly and attaches user info to the request
Should handle Bearer token format correctly	200	Confirms middleware correctly parses tokens with "Bearer" prefix

Authorization

Test Case	Expected Status	Notes
Should return 401 if user is not authenticated	401	Ensures access is denied when no user object is present in the request
Should return 403 if user role is not authorized	403	Restricts access to routes where user role doesn't match required roles
Should allow access if user role is authorized	200	Confirms authorized roles can access protected routes
Should allow access if user has one of the authorized roles	200	Checks that multiple allowed roles (e.g., admin/wholesaler) are properly validated
Should handle string role parameter	200	Verifies that a single role string is accepted instead of an array

Should allow access if no roles specified (any authenticated user)	200	Ensures that if no roles are required, any authenticated user can proceed

9 JWT Test (jwt.test.js)

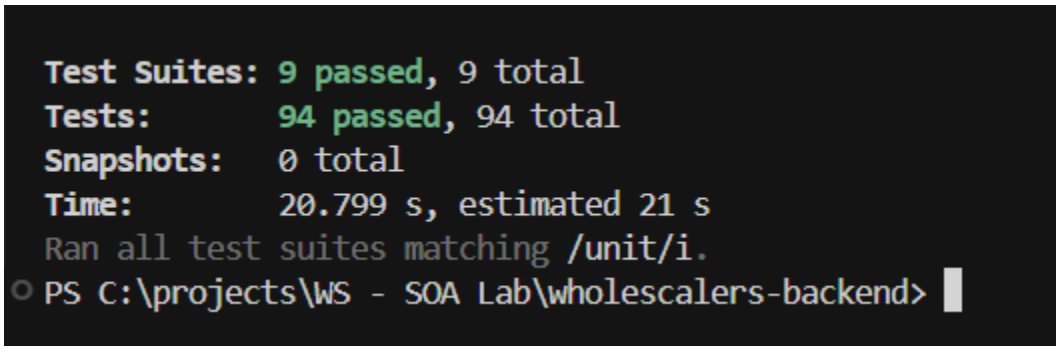
Sign Token

Test Case	Expected Status	Notes
Should create a valid JWT token	200	Ensures that the utility successfully generates a JWT string when provided a valid user object
Should include user id and role in token payload	200	Verifies that the signed token contains the correct user ID and role in its payload
Should handle different user roles	200	Confirms that the utility correctly encodes various user roles (retailer, wholesaler, admin) within the JWT

Running All Unit Tests

Suits : 9 Passed : 9

Tests : 94 Passed : 94



Integration Testing

Auth API (tests/integration/auth.test.js)

Endpoint	Method	Description	Status
/api/auth/register	POST	Register user	201, 400
/api/auth/login	POST	Login user	200, 400
/api/auth/me	GET	Get logged-in user	200, 401
/api/auth/logout	POST	Logout user	200

Products API (tests/integration/products.test.js)

Endpoint	Method	Description	Status
/api/products	GET	List all products	200, 401
/api/products?category=	GET	Filter products by category	200
/api/products/:id	GET	Get product by ID	200, 404
/api/products	POST	Create product	201, 401, 403
/api/products/:id	PUT	Update product	200
/api/products/:id	DELETE	Delete product	200

Orders API (tests/integration/orders.test.js)

Endpoint	Method	Description	Status
/api/orders	POST	Create new order	201, 400, 401
/api/orders	GET	List orders (retailer/wholesaler)	200
/api/orders/:id	GET	Get order details	200, 403
/api/orders/:id/status	PUT	Update order status	200, 403

Invoices API (tests/integration/invoices.test.js)

Endpoint	Method	Description	Status
/api/invoices/order/:orderId	POST	Create invoice for order	201, 403, 404
/api/invoices/:id	GET	Get invoice details	200, 404
/api/invoices	GET	List invoices (retailer)	200, 403
/api/invoices/wholesaler	GET	List invoices (wholesaler)	200, 403

Payments API (tests/integration/payments.test.js)

Endpoint	Method	Description	Status
/api/payments/create-order	POST	Create Razorpay order	200, 404, 403
/api/payments/verify-payment	POST	Verify Razorpay signature	200, 400, 404

Retailer Dashboard (tests/integration/retailer.test.js)

Endpoint	Method	Description	Status
/api/retailer/overview	GET	Retailer overview dashboard	200 (retailer), 403 (others)

Reports API (tests/integration/reports.test.js)

Endpoint	Method	Description	Status
/api/reports/sales	GET	Get sales report	200, 403
/api/reports/inventory	GET	Get inventory report	200, 403
/api/reports/customers	GET	Get customers report	200, 403

Settings API (tests/integration/settings.test.js)

Endpoint	Method	Description	Status
/api/settings/profile	GET	Get profile	200, 401
/api/settings/profile	PUT	Update profile	200
/api/settings/password	PUT	Change password	200, 400, 401

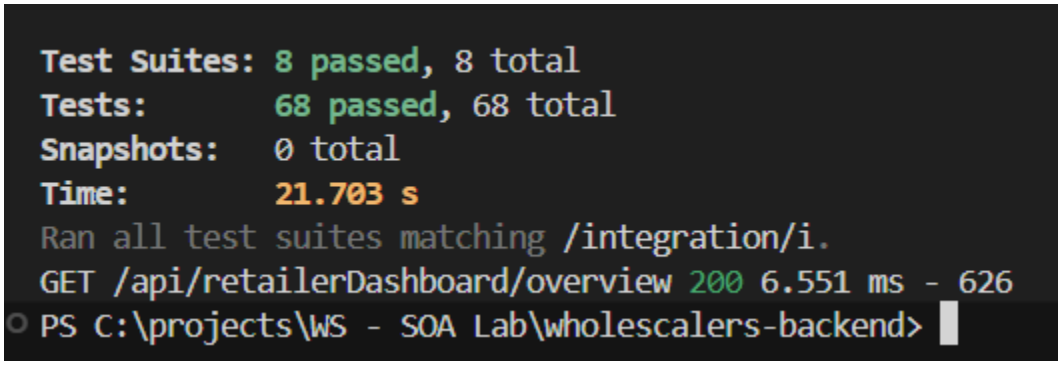
Common Issues & Fixes

Issue	Cause	Fix
404 errors	Missing routes	Check Express route mounts
401/403 errors	Missing or invalid JWT	Use getAuthHeaders(user)
MongoDB fails to start	Memory server error	Use fallback MONGODB_URI
Long teardown	Open DB handles	Ensure afterAll() closes connections

Running All Integration Tests

Suits : 8 Passed : 8

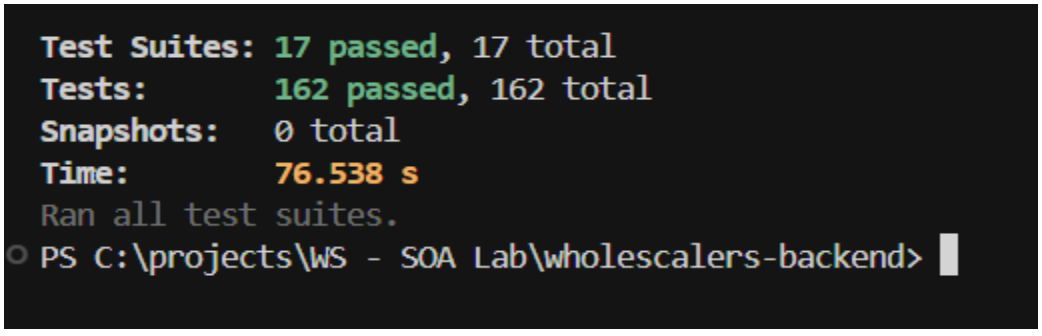
Tests : 68 Passed : 68



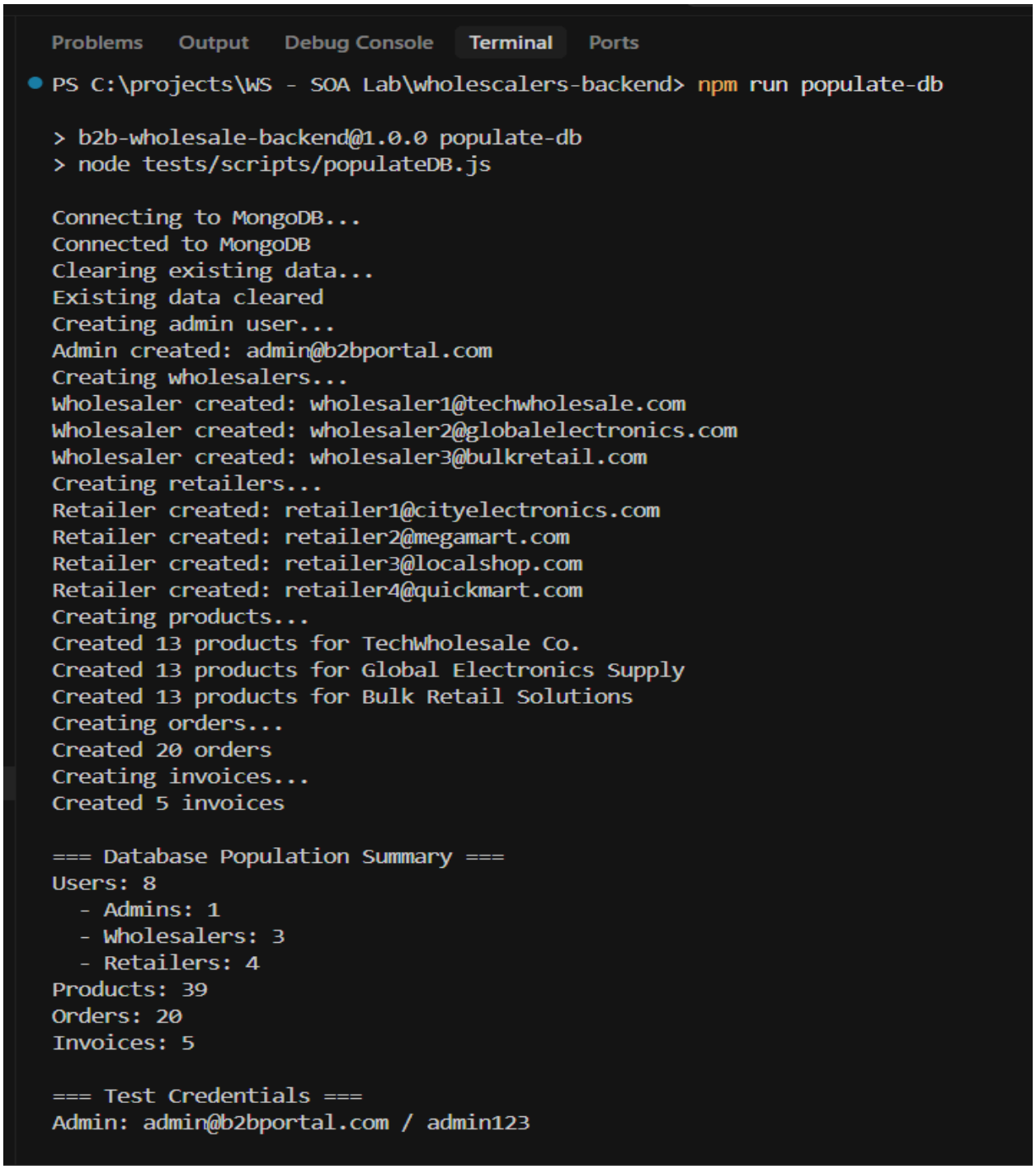
Running All Tests

Suits : 17 Passed : 17

Tests : 162 Passed : 162



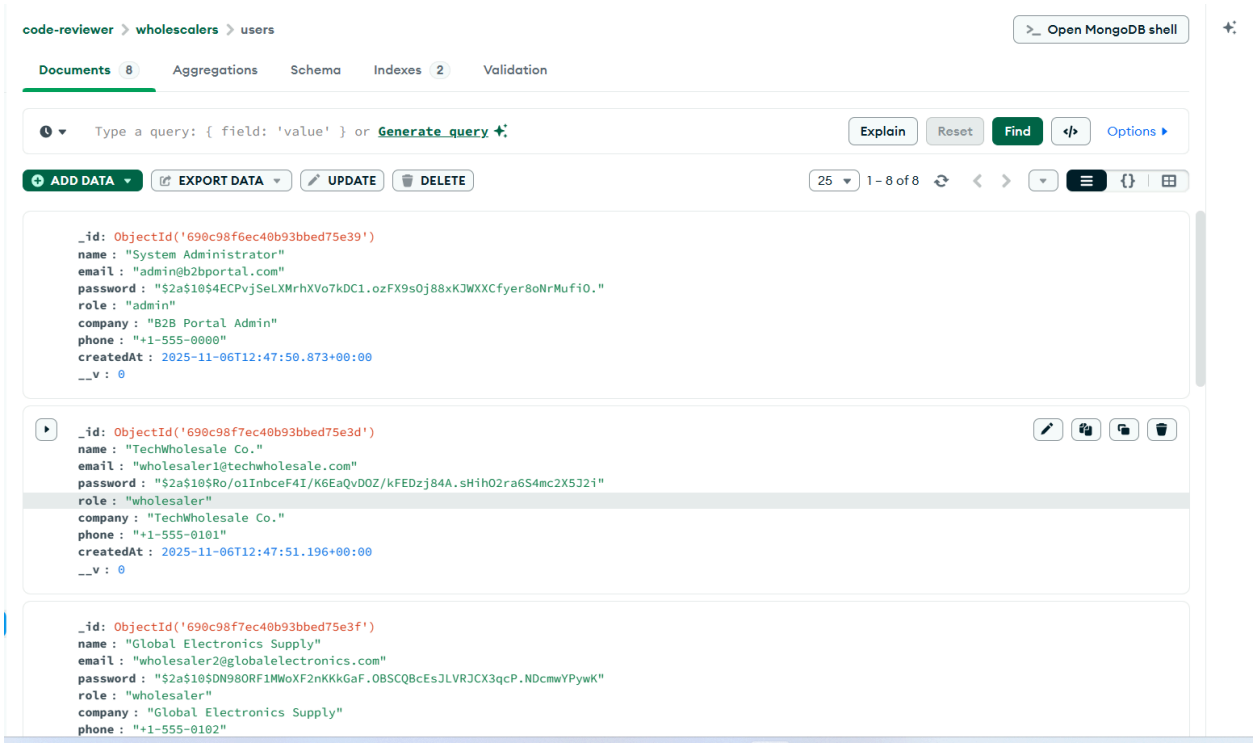
Populating DB

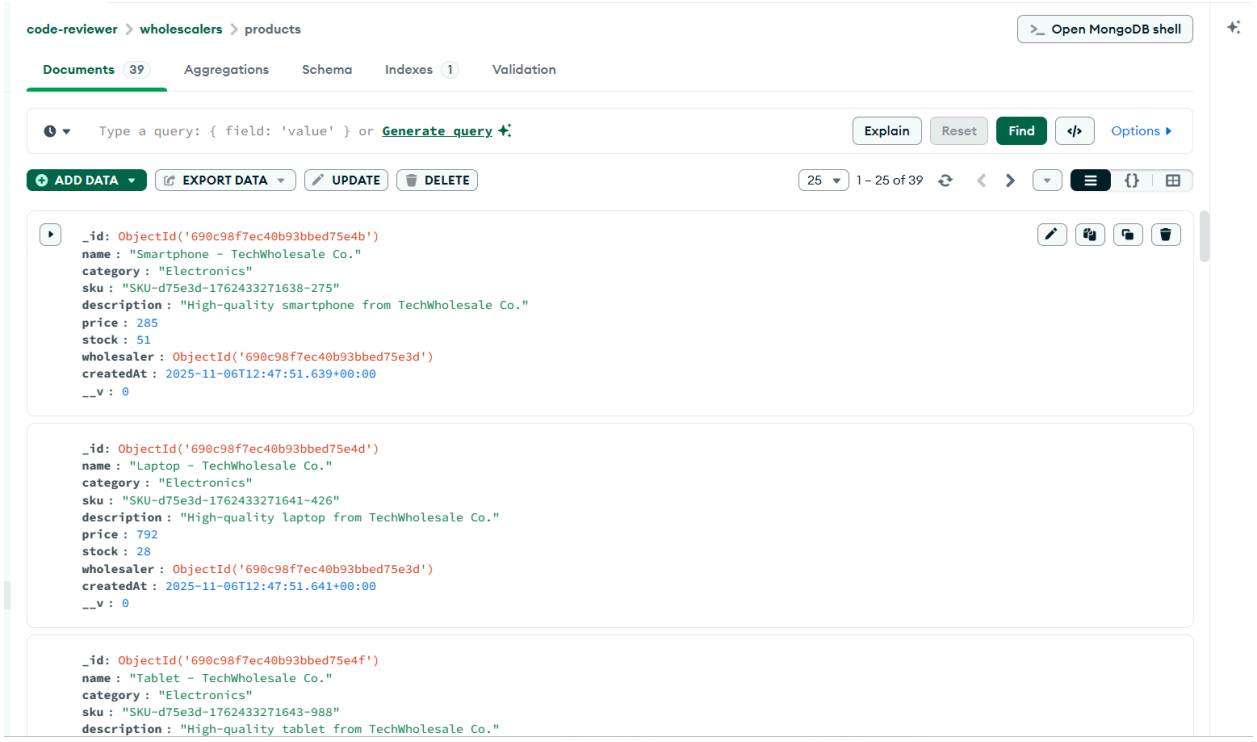


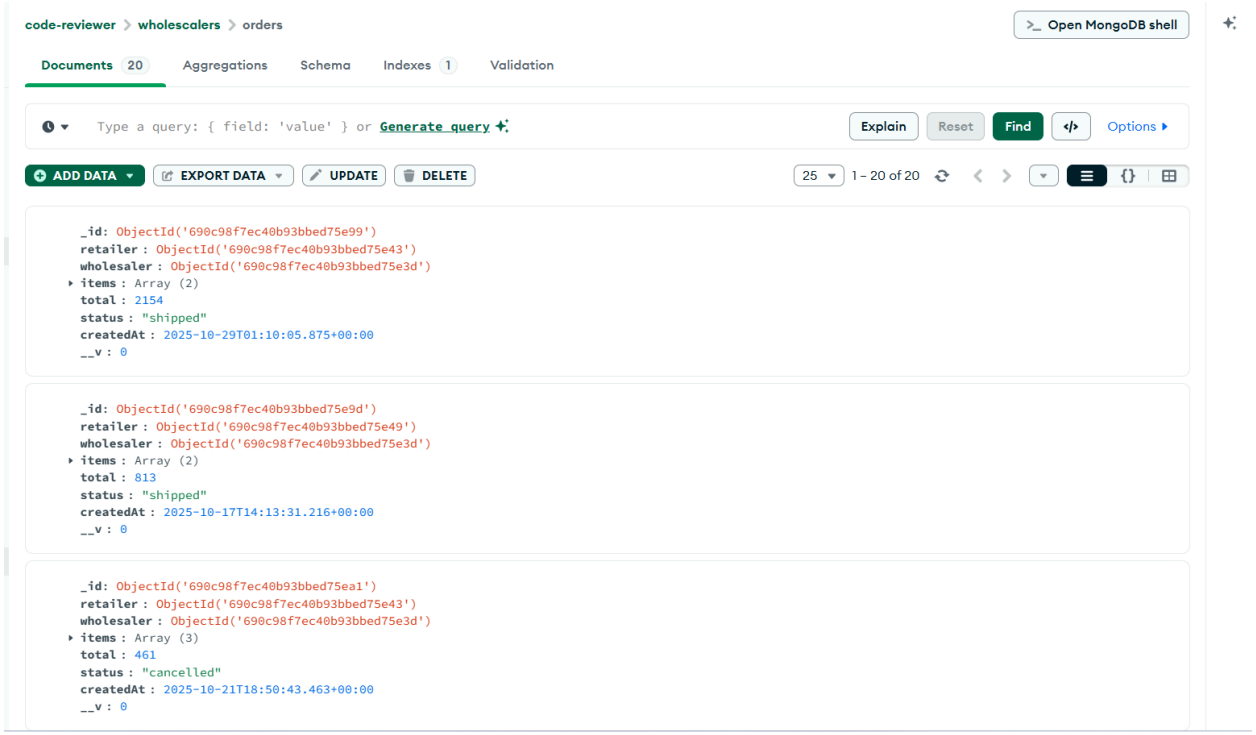
Problems Output Debug Console Terminal Ports

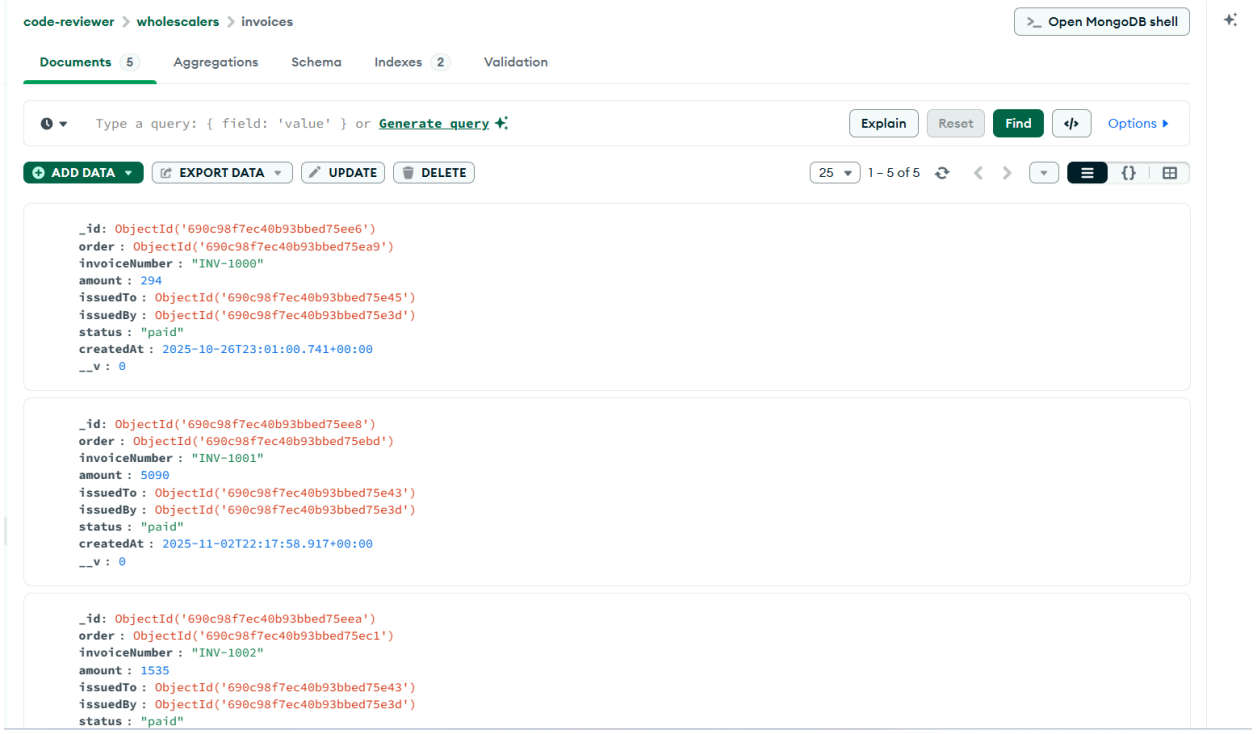
```
Connecting to MongoDB...
Connected to MongoDB
Clearing existing data...
Existing data cleared
Creating admin user...
Admin created: admin@b2bportal.com
Creating wholesalers...
Wholesaler created: wholesaler1@techwholesale.com
Wholesaler created: wholesaler2@globalelectronics.com
Wholesaler created: wholesaler3@bulkretail.com
Creating retailers...
Retailer created: retailer1@cityelectronics.com
Retailer created: retailer2@megamart.com
Retailer created: retailer3@localshop.com
Retailer created: retailer4@quickmart.com
Creating products...
Created 13 products for TechWholesale Co.
Created 13 products for Global Electronics Supply
Created 13 products for Bulk Retail Solutions
Creating orders...
Created 20 orders
Creating invoices...
Created 5 invoices

=== Database Population Summary ===
Users: 8
  - Admins: 1
  - Wholesalers: 3
  - Retailers: 4
Products: 39
Orders: 20
Invoices: 5

=== Test Credentials ===
Admin: admin@b2bportal.com / admin123
Wholesaler 1: wholesaler1@techwholesale.com / password123
Retailer 1: retailer1@cityelectronics.com / password123

Database populated successfully!
PS C:\projects\WS - SOA Lab\wholescalers-backend>
```

MongoDB Database :






code-reviewer > wholescalers > orders
Open MongoDB shell
Documents 20
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 – 20 of 20
_id: ObjectId('690c98f7ec40b93bbed75e99')
retailer : ObjectId('690c98f7ec40b93bbed75e43')
wholesaler : ObjectId('690c98f7ec40b93bbed75e3d')
items : Array (2)
total : 2154
status : "shipped"
createdAt : 2025-10-29T01:10:05.875+00:00
__v : 0
_id: ObjectId('690c98f7ec40b93bbed75e9d')
retailer : ObjectId('690c98f7ec40b93bbed75e49')
wholesaler : ObjectId('690c98f7ec40b93bbed75e3d')
items : Array (2)
total : 813
status : "shipped"
createdAt : 2025-10-17T14:13:31.216+00:00
__v : 0
_id: ObjectId('690c98f7ec40b93bbed75ea1')
retailer : ObjectId('690c98f7ec40b93bbed75e43')
wholesaler : ObjectId('690c98f7ec40b93bbed75e3d')
items : Array (3)
total : 461
status : "cancelled"
createdAt : 2025-10-21T18:50:43.463+00:00
__v : 0


code-reviewer > wholescalers > invoices
Open MongoDB shell
Documents 5
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 – 5 of 5
_id: ObjectId('690c98f7ec40b93bbed75ee6')
order : ObjectId('690c98f7ec40b93bbed75ea9')
invoiceNumber : "INV-1000"
amount : 294
issuedTo : ObjectId('690c98f7ec40b93bbed75e45')
issuedBy : ObjectId('690c98f7ec40b93bbed75e3d')
status : "paid"
createdAt : 2025-10-26T23:01:00.741+00:00
__v : 0
_id: ObjectId('690c98f7ec40b93bbed75ee8')
order : ObjectId('690c98f7ec40b93bbed75ebd')
invoiceNumber : "INV-1001"
amount : 5090
issuedTo : ObjectId('690c98f7ec40b93bbed75e43')
issuedBy : ObjectId('690c98f7ec40b93bbed75e3d')
status : "paid"
createdAt : 2025-11-02T22:17:58.917+00:00
__v : 0
_id: ObjectId('690c98f7ec40b93bbed75eea')
order : ObjectId('690c98f7ec40b93bbed75ec1')
invoiceNumber : "INV-1002"
amount : 1535
issuedTo : ObjectId('690c98f7ec40b93bbed75e43')
issuedBy : ObjectId('690c98f7ec40b93bbed75e3d')
status : "paid"