



islington college
(इस्लिंग्टन कॉलेज)

Module Code & Module Title

CS6P05NI Final Year Project

Assessment Weightage & Type

40% FYP Final Report

Semester

202 Autumn

PROJECT TITLE: Udyog

Student Name:

London Met ID:

College ID:

Internal Supervisor:

External Supervisor:

Assignment Due Date:

Assignment Submission Date:

Word Count : 8788

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

I would like to take this opportunity to express my deepest gratitude to my family for their unwavering love and support throughout my academic journey. Their constant encouragement and motivation have been invaluable to me, and I couldn't have reached this far without them.

I would also like to extend my heartfelt thanks to Islington College and FYP Support for providing me with the opportunity to conduct this project. Their guidance and support have been instrumental in shaping this project and I am truly grateful for their contribution.

Furthermore, I would like to acknowledge the immense help and support provided by my supervisors, Sir and Sir. Their constant guidance, constructive feedback, and willingness to help me overcome any obstacles I faced during the project have been truly remarkable. I owe a great deal of my success to them, and I am humbled by their dedication and professionalism.

Once again, I express my sincere appreciation to all those who have contributed to this project and helped me achieve this milestone in my academic journey.

Abstract

This report outlines the format of an academic project that aims to address the problems with conventional data storage and stock management methods that businesses in Nepal encounter. The conventional method wastes important human resources since it is unreliable, difficult to maintain, and unsafe. Furthermore, just 50% of organizations worldwide employ big data to expand their operations, and the industry's digitization strategy fails to turn data into information that is usable. A survey is used to identify the project's problem domain, which includes pricey software, features that are unrelated to the Nepali environment, date conversion, and human error that results in financial loss. The introduction, background, development, testing, and analysis parts are all included in the report's format. An overview of comparable projects used in the market, as well as the tools and technology employed to finish the project, are provided in the background section. The development area includes information on the selected methodology, activities completed, development coding, requirement analysis, feedback reviews, and automatically generated test cases produced sprint-by-sprint. The section on testing and analysis includes the many test cases, test situations, and necessary adjustments made. The conclusion part examines the built application's shortcomings, benefits to users, upcoming features to enhance user experience, obstacles encountered, and solutions adopted to address shortcomings and challenges. This research stresses the significance of tackling traditional data storage difficulties for businesses in Nepal and offers a thorough framework for an academic project.

Table of Contents

1.	Introduction	1
1.1.	Project Overview.....	2
1.2.	Current Scenario.....	3
1.2.1.	In context of Nepal.....	3
1.3.	Problem Domain	4
1.3.1.	Problem Domain Listed from research.....	4
1.3.2.	Problem Domain Listed from survey.....	4
1.4.	Project Scope.....	5
1.5.	Aims and Objectives	6
1.5.1.	Aim of the Project	6
1.5.2.	Objectives of the Project.....	6
1.6.	Structure of the report.....	7
1.6.1.	Introduction	7
1.6.2.	Background.....	7
1.6.3.	Development.....	7
1.6.4.	Testing and analysis	7
1.6.5.	Conclusion	8
2.	Background.....	9
2.1.	Targeted User Type	9
2.2.	End User.....	9
2.2.1.	Client Information.....	9
2.3.	Understanding the solution	10
2.3.1.	Solution for Client.....	10
2.4.	Review of similar project	11

2.4.1.	Swastik Business Accounting Software	11
2.4.2.	Vyapar App.....	12
2.4.3.	Tally	13
2.5.	Comparison Between System.....	14
3.	Development.....	15
3.1.	Considered Methodology	15
3.1.1.	Waterfall Model.....	15
3.1.1.1.	Reason for not choosing Waterfall Model	15
3.1.2.	Prototype Methodology	16
3.1.2.1.	Reason not choosing Prototype Model	16
3.1.3.	Spiral Model.....	17
3.1.3.1.	Reason not choosing Spiral Model	17
3.1.4.	Feature Driven Development (FDD)	18
	Brief discussion about FDD is included in appendix section.....	18
3.1.4.1.	Reason not choosing FDD Model	18
3.2.	Selected Methodology	19
3.2.1.	About Scrum	19
3.2.2.	Reason choosing SCRUM.....	20
3.3.	Phases of Methodology and its implementation.....	21
3.3.1.	Initiation	21
3.3.2.	Planning and Estimation	21
3.3.3.	Implementation	21
3.3.4.	Reviewing	22
3.3.5.	Releasing.....	22
3.4.	Survey Results.....	23

3.4.1.	Highlights from Pre-Survey	23
3.4.2.	Highlights from Post-Survey	23
3.5.	Requirement Analysis	24
3.5.1.	Software Requirements	24
3.5.1.1.	Notable Packages used.....	25
3.6.	Design.....	26
3.6.1.	Application Logo	26
3.6.2.	System Design.....	27
3.6.2.1.	Use Case Diagram	27
3.6.2.2.	ER Diagram	28
3.6.2.3.	Gantt Chart	29
3.6.2.4.	Activity Diagram.....	30
3.6.2.5.	DFD Diagram.....	34
3.6.3.	Feature Design	35
3.6.3.1.	Register User.....	35
3.6.3.2.	Login User	37
3.6.3.3.	Generate Invoices.....	39
3.6.3.4.	Add Packages to Invoices	41
3.6.3.5.	Mark Attendance for employee.....	42
3.6.3.6.	View Sales Report	44
3.6.3.7.	Importing data from CSV file.....	46
3.6.3.8.	Adding Employee.....	48
3.6.3.9.	Generating QR Code for Packages	50
3.6.4.	Wireframes Developed	52
3.7.	Implementation	77

3.7.1.	Sprint 1 – User Management	77
3.7.1.1.	Sprint Planning	77
3.7.1.2.	Development	79
3.7.1.3.	Testing.....	83
3.7.1.4.	Sprint Retrospective	84
3.7.2.	Sprint 2 – Inventory Management System	85
3.7.2.1.	Sprint Planning	85
3.7.2.2.	Development	88
3.7.2.3.	Testing.....	93
3.7.2.4.	Sprint Retrospective	94
3.7.3.	Sprint 3 – Invoice Billing System.....	95
3.7.3.1.	Sprint Planning	95
3.7.3.2.	Development	102
3.7.3.3.	Testing.....	115
3.7.3.4.	Sprint Retrospective	115
3.7.4.	Sprint 4 – Employee Management System.....	116
3.7.4.1.	Sprint Planning	116
3.7.4.2.	Development	116
3.7.4.3.	Testing.....	123
3.7.4.4.	Sprint Retrospective	124
3.7.5.	Sprint 5 – Role Management System	125
3.7.5.1.	Sprint Planning	125
3.7.5.2.	Development	125
3.7.5.3.	Sprint Retrospective	133
3.7.6.	Sprint 6 – Search	134

3.7.6.1.	Sprint Planning	134
3.7.6.2.	Development	134
3.7.6.3.	Testing.....	137
3.7.6.4.	Sprint Retrospective	137
3.7.7.	Sprint 7 – Notifications.....	138
3.7.7.1.	Sprint Planning	138
3.7.7.2.	Development	140
3.7.7.3.	Testing.....	143
3.7.7.4.	Sprint Retrospective	144
3.7.8.	Sprint 8 - QR Code Implementation.....	145
3.7.8.1.	Sprint Planning	145
3.7.8.2.	Development	148
3.7.8.3.	Sprint Retrospective	151
3.7.9.	Sprint 9 – Import / Export Files	152
3.7.9.1.	Sprint Planning	152
3.7.9.2.	Development	155
3.7.9.3.	Testing.....	157
3.7.9.4.	Sprint Retrospective	158
3.7.10.	Sprint 10 – Report Generation	159
3.7.10.1.	Sprint Planning	159
3.7.10.2.	Development	161
3.7.10.3.	Testing.....	163
3.7.10.4.	Sprint Retrospective	164
4.	Testing and analysis	165
4.1.	Testing	165

4.1.1.	Test 1 - Login Testing with Invalid login credentials.....	165
4.1.2.	Test 2 - Login in the system with valid login credentials	166
4.1.3.	Test 3 - Testing form for possibilities of SQL injection	168
4.1.4.	Test 4 - Adding new user with existing username.....	169
4.1.5.	Test 5 - Adding new user with valid details.....	170
4.1.6.	Test 6 - API testing for protected routes.	171
4.1.7.	Test 7 - Testing to add products pages with existing product id.\	172
4.1.8.	Test 8 - Testing to add products pages with valid details.....	173
4.1.9.	Checking whether location of the of customer update on selecting customer.	
	174	
4.1.10.	Test 10 - Adding empty invoice.	175
4.1.11.	Test 11 - Changing Employee Status.....	176
4.1.12.	Test 12 - Checking frontend validation for adding employee.....	177
4.1.13.	Test 13 - Adding employee attendance.....	178
4.1.14.	Test 14 - Adding duplicate attendance entry of all employees.	179
4.1.15.	Test 15 - Adding new employee record.....	180
4.1.16.	Test 16 - Testing the functionality of data being inserted in elastic search indexes.	181
4.1.17.	Test 17 - API testing for searching data.	182
4.1.18.	Test 18 - API testing for getting all user notifications.....	183
4.1.19.	Test 19: Checking whether Invoice generated are being sent to the customer email.....	184
4.1.20.	Test 20 - API testing for getting supervisor dashboard report.	185
4.1.21.	Test 21 - Testing whether new sales made will be received in the dashboard API.	186
4.1.22.	Test 22 - Testing API for generating reports.	187

4.1.23.	Test 23 - Testing for downloading data in PDF format.	188
4.1.24.	Test 24: Testing for downloading data in CSV format.	190
4.1.25.	Test 25 - Uploading data from CSV format.	191
4.1.26.	Test 26 - API testing for generating packages with products.	192
4.1.27.	Test 27 - Testing to generate QR code for production package.	194
4.1.28.	Test 28: Testing to add products from package to the invoice.	195
4.1.29.	Test 29: Testing to add same package multiple times.	196
4.1.30.	Test 30: Testing authorized route to access users API as supervisor.	197
4.1.31.	Test 31: Performing same test above but as admin.	199
4.1.32.	Test 32: Testing whether React redirect to correct portal based on user type.	201
4.1.33.	Test 33: Testing to access supervisor portal logged in as staff.	204
4.1.34.	Test 34: Testing whether staff pages are displayed as per assigned role.	205
4.1.35.	Test 35: Testing Employee Attendance Cron Job	207
4.1.36.	Test 36: Testing Low Stock Alert cron job for supervisors.	208
4.1.37.	Test 37: Testing Overdue Alert for all customers.	210
4.2.	Critical Analysis.....	211
4.2.1.	Analysis on selected methodology.....	211
4.2.2.	Analyzing failed test cases.....	212
4.2.2.1.	Testing SQL Injection	212
4.2.2.2.	Sales breakdown by day testing	213
4.2.3.	Test Summary	214
5.	Conclusion	215
5.1.	Legal, Social and Ethical Issues	215
5.1.1.	Legal Issues.....	215

5.1.1.1.	Unregistered Software	215
5.1.1.2.	Collection of Personal Information	215
5.1.2.	Social Issues.....	216
5.1.2.1.	Privacy Concern	216
5.1.2.2.	Intolerance of Change	216
5.1.2.3.	Workplace exploitation.....	216
5.1.3.	Ethical Issues.....	217
5.1.3.1.	Job Displacement	217
5.1.3.2.	Digital Divide.....	217
5.2.	Advantages	218
5.2.1.	Improved Efficiency.....	218
5.2.2.	Better Inventory Management.....	218
5.2.3.	Employee Management	218
5.2.4.	Data Analytics and Reporting	218
5.3.	Limitations.....	219
5.3.1.	No Employee Progress Tracking	219
5.3.2.	Unable to Maintain Batch Wise Production.....	219
5.3.3.	Only Applicable for Single Warehouse.....	219
5.3.4.	Manual Employee Attendance	219
5.4.	Future Work	220
5.4.1.	Registering the application with IRD	220
5.4.2.	Batch & Warehouse Wise Production Tracking	220
5.4.3.	Customer Risk Analysis	220
5.4.4.	Mobile Application Development.....	221
5.4.5.	Integration of the management system with IoT devices.	221

5.4.6.	Employee Performance Monitoring.....	221
5.4.7.	Change the product from CPaaS to SaaS	221
6.	References.....	222
7.	Appendix	225
7.1.	Appendix A – Pre-Survey (Sample)	225
7.2.	Appendix B – Design	229
7.2.1.	Gantt Chart Development	229
7.2.2.	Use Case Development.....	230
7.2.2.1.	First rehearsal of use case diagram.....	230
7.2.3.	ERD Development	231
7.2.3.1.	First Rehearsal of ERD.....	231
7.2.3.2.	Second Rehearsal of ERD.....	232
7.3.	Appendix C - User Feedback.....	233
7.3.1.	Form Sample	233
7.3.2.	Post Survey Result	236
7.3.3.	Comparison of Result from post survey and pre survey	239
7.4.	Appendix D – Similar Projects.....	240
7.4.1.	Zoho Inventory	240
7.5.	Appendix E – Product Backlog Changes	241
7.5.1.	Changes made in Product Backlog.....	241
7.5.2.	Final Product Backlog	241
7.6.	Appendix F – Considered Methodology	242
7.6.1.	FDD Model.....	242
7.6.1.1.	Lifecycle/Phases of FDD Model.....	242
7.6.1.2.	Advantages of FDD Model.....	243

7.6.1.3. Roles and Responsibilities.....	243
7.6.2. Spiral Model.....	244
7.6.2.1. Lifecycle/Phases of Spiral Model	244
7.6.2.2. Advantages of Spiral Model	244
7.6.2.3. Roles and Responsibilities.....	245
7.6.3. Prototype Model.....	246
7.6.3.1. Lifecycle/Phases of Prototype Model.....	246
7.6.3.2. Roles and Responsibilities.....	247
7.6.3.3. Advantages of Prototype Model.....	247
7.6.4. Waterfall Model.....	248
7.6.4.1. Lifecycle/Phases of Waterfall Model	248
7.6.4.2. Roles and Responsibilities.....	249
7.6.4.3. Advantages of Waterfall.....	249
7.7. Appendix I – Tools and Technology Used.....	250
7.7.1. React JS	250
7.7.2. HTML / CSS (Tailwind CSS).....	251
7.7.3. Node JS.....	252
7.7.4. PostgreSQL	253
7.7.5. Elastic Search.....	254
7.8. SRS Document	255
7.8.1. Introduction	255
7.8.1.1. Purpose of Document	255
7.8.1.2. Scope of the Document	255
7.8.1.3. Overview.....	255
7.8.2. Overall Description.....	256

7.8.2.1.	Project Perspective.....	256
7.8.2.2.	User Classes and characteristics.....	257
7.8.2.3.	Operating environment	257
7.8.2.4.	Design and Implementation constraints.....	257
7.8.2.5.	User Documentation	258
7.8.2.6.	System Features.....	258
7.8.2.7.	Functional Requirements	260
7.8.2.8.	Other Non-Functional Requirements	270
7.9.	Important Coding Logic.....	271
7.9.1.	Cron Jobs In Application.....	271
7.9.2.	Global Mail Function	273
7.10.	Git according to Sprint.....	279
7.11.	Originality Report.....	280

List of Figures

Figure 1: Screenshot of Swastik App	11
Figure 2: Vaypar App	12
Figure 3: Sprint Cycle with its sub process.....	19
Figure 4: Application Logo.....	26
Figure 5: Use case diagram	27
Figure 6: Final ER Diagram for project	28
Figure 7: Final Gantt Chart.....	29
Figure 8: Activity Diagram for registering new user.....	30
Figure 9: Activity diagram for accessing dashboard.....	31
Figure 10: Activity diagram for adding new employee.....	32
Figure 11: Activity Diagram for user setting.....	33
Figure 12: Context Level Diagram for Udhyyog	34
Figure 13: Level 1 DFD for Udhyyog.....	34
Figure 14:DFD Level 2 for registering new user.....	35
Figure 15: Collaboration Diagram for registration.....	35
Figure 16: Sequence Diagram for registering new user.	36
Figure 17: DFD Level 2 for login user.....	37
Figure 18: Communication Diagram for Login.....	37
Figure 19: Sequence Diagram for Login.....	38
Figure 20: DFD Level 2 for generating invoices.	39
Figure 21: Communication diagram for generating invoices.	39
Figure 22:Sequence diagram for generating invoices	40
Figure 23: DFD Level 2 for adding packages to invoices.	41
Figure 24: Communication diagram for adding packages to invoices.	41
Figure 25:DFD Level 2 for adding attendance for employee.	42
Figure 26: Communication diagram for adding attendance for employee	42
Figure 27:Sequences diagram for adding attendance for employee.	43
Figure 28: DFD Level 2 for generating sales report.....	44
Figure 29: Communication diagram for generating sales report.....	44
Figure 30: Sequence diagram for generating sales report.	45

Figure 31: DFD Level 2 for importing data from csv.....	46
Figure 32:Communication diagram for importing data form csv.....	46
Figure 33: Sequence diagram for importing data for csv.....	47
Figure 34: DFD Level 2 for adding new employees.....	48
Figure 35: Collaboration Diagram for adding employee.....	48
Figure 36: Sequence diagram for adding employee.....	49
Figure 37: DFD Level 2 for generating QR Code.....	50
Figure 38: Communication Diagram for generating QR Code.....	50
Figure 39: Sequence diagram for generating QR code.....	51
Figure 40: Add Employee Wireframe	52
Figure 41: Add Invoice Wireframe	53
Figure 42: Add Package Wireframe	54
Figure 43: Add Transaction Wireframe.....	55
Figure 44: Add Customer Wireframe.....	56
Figure 45: Add Products Page	57
Figure 46: Admin Dashboard Wireframe	58
Figure 47: Customer Wireframe	59
Figure 48: Edit Customer Wireframe	60
Figure 49: Edit Invoice Wireframe	61
Figure 50: Edit Product Wireframe	62
Figure 51: Edit Role WIreframe	63
Figure 52:Edit User Wireframe	64
Figure 53: Add Attendance Wireframe	65
Figure 54: Employee Wireframe	66
Figure 55: Invoice Wireframe	67
Figure 56: Login Wireframe	68
Figure 57: Production Wireframe.....	69
Figure 58: Product Wireframe	70
Figure 59: Generate Report Wireframe	71
Figure 60: Role Wireframe	72
Figure 61: Supervisor Dashboard Wireframe	73

Figure 62: Transaction Wireframe	74
Figure 63: Users Wireframe	75
Figure 64: Package Info Wireframe.....	76
Figure 65: Figma Mockup for Login Page	77
Figure 66: Figma Mockup for registering new user	78
Figure 67: Table Design for user's table.....	79
Figure 68: Developed Frontend UI from mockup for adding users.....	80
Figure 69: Screenshot of code developed for adding user.....	81
Figure 70: Developed Frontend UI from mockup for Login Page.	82
Figure 71: Screenshot of code developed for logging in.	82
Figure 72: Screenshot of API testing result for Sprint 1.....	83
Figure 73: Screenshot of automatic testing result for react components.....	83
Figure 74: Figma Mockup for Products Page	85
Figure 75: Figma Mockup for Add Product Page	86
Figure 76: Figma Mockup for Edit Product Page.....	87
Figure 77: Table Design for product's table.....	88
Figure 78: Table Design for product category table	89
Figure 79: Developed Frontend UI from mockup for viewing product list.	90
Figure 80: Screenshot of code developed for viewing product list.....	90
Figure 81: Developed Frontend UI from mockup for adding product.....	91
Figure 82: Screenshot of code developed for adding new product.....	91
Figure 83: Developed Frontend UI from mockup for editing product.	92
Figure 84: Screenshot of code developed for editing product.	92
Figure 85: Screenshot of API testing result for Sprint 2.....	93
Figure 86: Screenshot of automatic test result for react components.....	93
Figure 87: Figma Mockup for Invoice Page	95
Figure 88: Figma Mockup for Add Invoice Page	96
Figure 89: Figma Mockup for Add Transaction Page	97
Figure 90: Figma Mockup for Transaction Page	98
Figure 91: Figma Mockup for Customer Information Page.....	99
Figure 92: Figma Mockup for Customer Page.....	100

Figure 93: Figma Mockup for Add Customer Page	101
Figure 94: Table Design for customer's table.....	102
Figure 95: Table Design for Nepal's district table	103
Figure 96: Table Design for money transaction table	103
Figure 97: Table Design for invoices table	104
Figure 98: Table Design for invoices product table	104
Figure 99: Table Design for product transaction table.....	105
Figure 100: Screenshot of developed invoice page.....	106
Figure 101: Screenshot of code developed for viewing invoices list.....	106
Figure 102: Screenshot of developed add invoice page.....	107
Figure 103: Screenshot of code developed for adding invoices.	107
Figure 104: Screenshot of developed edit invoice page.....	108
Figure 105: Screenshot of code developed for editing invoices.	108
Figure 106: Screenshot of developed add customer page.	109
Figure 107: Screenshot of code developed for adding customer.	109
Figure 108: Screenshot of developed customer page.	110
Figure 109: Screenshot of code developed for viewing customer list.....	110
Figure 110: Screenshot of developed customer information page.	111
Figure 111: Screenshot of code developed for viewing customer information.	111
Figure 112: Screenshot of developed edit customer page.	112
Figure 113: Screenshot of code developed for editing customers.....	112
Figure 114: Screenshot of developed transaction page.	113
Figure 115: Screenshot of code developed for viewing transaction list.	113
Figure 116: Screenshot of developed add transaction page.	114
Figure 117: Screenshot of code developed for adding transaction.....	114
Figure 118: Screenshot of API testing result	115
Figure 119: Screenshot of frontend testing result.....	115
Figure 120: Figma mockup for adding attendance page.	116
Figure 121: Figma mockup for adding employee.	117
Figure 122: Figma mockup for viewing employee information.	118
Figure 123: Figma mockup for viewing employee list page.....	119

Figure 124: Screenshot of developed employee list page.....	120
Figure 125: Screenshot of developed code for viewing employee list.	120
Figure 126: Screenshot of developed code for adding new employee.	121
Figure 127: Screenshot of developed code for adding new employees.	121
Figure 128: Screenshot of developed employee profile page	122
Figure 129: Screenshot of developed code for employee profile page.....	122
Figure 130: Screenshot of test case result developed for Sprint 4	123
Figure 131: Screenshot of success result from cypress test case.....	123
Figure 132: Initially created role-based system.	125
Figure 133: Initially planned ERD for role management system.....	126
Figure 134: Figma mockup for creating user.....	127
Figure 135: Table design for Sprint 5 - role_with_permission	128
Figure 136: Table design for Sprint 5 - users	128
Figure 137: Table design for Sprint 5 - permission_lists.....	129
Figure 138: Screenshot of developed add user page.....	130
Figure 139: Screenshot of developed code for add user page.....	130
Figure 140: Screenshot of developed user page.....	131
Figure 141: Screenshot of developed code for viewing user list.	131
Figure 142: Screenshot of developed edit user page.....	132
Figure 143: Screenshot of developed code for edit user page	132
Figure 144: Figma mockup generated for search result.	134
Figure 145:Screenshot of developed search input field.....	135
Figure 146: Screenshot of developed code for search input field.....	135
Figure 147: Screenshot of developed search result page	136
Figure 148: Screenshot of code developed for search result.	136
Figure 149: Screenshot of test case result developed for Sprint 6	137
Figure 150: Successful test case for frontend components.....	137
Figure 151: Figma mockup for view all notification page.....	138
Figure 152: Figma mockup for preview of notification	139
Figure 153: Table design for notification table.....	140
Figure 154: Table design for user notification status table	140

Figure 155: Screenshot of developed preview of notification	141
Figure 156: Screenshot of developed code for notification preview	141
Figure 157: Screenshot of developed all notification page.....	142
Figure 158: Screenshot of developed code for viewing all notifications.	142
Figure 159: Test case result developed for Sprint 7	143
Figure 160: Successful test case for frontend components.....	143
Figure 161: Figma Mockup for adding production.	146
Figure 162: Figma Mockup for production.....	147
Figure 163: Figma mockup for QR code.	147
Figure 164: Developed production list page	148
Figure 165: Developed code for viewing production.....	148
Figure 166: Developed add production page.	149
Figure 167: Developed code for add production.	149
Figure 168: Developed QR code modal.	150
Figure 169: Developed code for QR Code modal	150
Figure 170: Figma Mockup for menu design for downloading.....	153
Figure 171: Figma Mockup for uploading CSV file.	154
Figure 172: Developed CSV upload component	155
Figure 173: Code developed for uploading CSV.	156
Figure 174: Developed download file menu	156
Figure 175: Successful test case for frontend components.....	157
Figure 176: Successful test case for frontend components.....	157
Figure 177: Figma Mockup for admin dashboard.....	159
Figure 178: Figma Mockup for Supervisor dashboard.....	160
Figure 179: Developed supervisor dashboard.....	161
Figure 180: Code developed for supervisor dashboard.....	162
Figure 181: Test case result developed for Sprint 10	163
Figure 182: Successful test case for frontend components.....	163
Figure 183: Error message for logging with invalid credentials	166
Figure 184: Success message for logging with valid credentials	167
Figure 185: User being redirected to respective dashboard.....	167

Figure 186: Error message in console.....	168
Figure 187: Error message on existing username.....	169
Figure 188: User being added successfully.....	170
Figure 189: Error message thrown for unauthorized access	171
Figure 190: Error message thrown for existing id.....	172
Figure 191: Success message displayed for adding new product.....	173
Figure 192: Choosing a customer from dropdown.	174
Figure 193: Location changed after choosing a customer.....	174
Figure 194: Error message being displayed.....	175
Figure 195: Employee status being successfully updated.....	176
Figure 196: Error message being displayed.....	177
Figure 197: Screenshot of success message for adding attendance.	178
Figure 198: Screenshot of error message being displayed for duplicate entry	179
Figure 199: Screenshot of employee being added to the application.....	180
Figure 200: Screenshot of adding employees.	181
Figure 201: Screenshot of Kibana response for searching added employees.	181
Figure 202: Screenshot of search result for requested keyword.	182
Figure 203:Screenshot of successful response for user notification.....	183
Figure 204:Screenshot of sent email preview where customer received the invoice... ..	184
Figure 205: Screenshot of API response for dashboard.....	185
Figure 206: Screenshot of incorrectly formatted API response	186
Figure 207: Screenshot of API response for report requested.	187
Figure 208: Screenshot of downloading in PDF format.	188
Figure 209: Screenshot of wrongly added customers.	189
Figure 210: Screenshot of downloading customer in correct format.....	189
Figure 211: Screenshot of downloading in CSV format.	190
Figure 212: Screenshot of downloaded CSV file.....	190
Figure 213: Screenshot of data been successfully added from csv file.	191
Figure 214: Screenshot of success response from API.....	193
Figure 215: Screenshot for generating QR Code for a package.	194
Figure 216: Screenshot of QR code generated for a package.	194

Figure 217: Screenshot of scanning QR.	195
Figure 218: Screenshot of package details from above scanned QR	195
Figure 219: Screenshot of error message being displayed.	196
Figure 220: Screenshot of logging in as supervisor	197
Figure 221: Screenshot of error message thrown for accessing unauthorized API	198
Figure 222: Screenshot of logging in as admin	199
Figure 223: Screenshot of success response for authorized API	200
Figure 224: Screenshot of logging in as supervisor	201
Figure 225: Screenshot of user redirected to supervisor portal	202
Figure 226: Screenshot of logging in as admin	202
Figure 227: Screenshot of user redirected to admin dashboard	203
Figure 228: Screenshot of 404 page for accessing unauthorized route.	204
Figure 229: Screenshot of staff available permission	205
Figure 230: Screenshot of available navlink based on user permission	206
Figure 231: Screenshot of data entry for today's attendance	207
Figure 232: Screenshot of cron log with success	207
Figure 233: Mail Format for low stock alert	208
Figure 234: Screenshot of PDF received in mail.	209
Figure 235: Screenshot for list of email send to all customers.	210
Figure 236: Mail format send to customer.	210
Figure 237: Screenshot of login code after making changes	212
Figure 238: Screenshot of actual result after changes in code	212
Figure 239: Code fix to solve the date format problem.	213
Figure 240: Screenshot of expected result after code fix	213
Figure 241: Processes required for registration of CMS	220
Figure 242: Survey Form Sample (2)	225
Figure 243: Survey Form Sample (2)	227
Figure 244: Survey Form Sample (3)	227
Figure 245: Survey Form Sample (4)	228
Figure 246: Gantt Chart rehearsals	229
Figure 247: Use Case first rehearsals	230

Figure 248: First Iteration of ERD.....	231
Figure 249: Second Iteration of ERD.....	232
Figure 250: User Feedback Form (1)	233
Figure 251: User Feedback Form (2)	234
Figure 252: User Feedback Form (3)	235
Figure 253: User Feedback Form (4)	235
Figure 254: Post survey result (1)	236
Figure 255: Post survey result (2)	236
Figure 256: Post survey result (3)	237
Figure 257: Post survey result (4)	237
Figure 258: Post survey result (5)	238
Figure 259: Post survey result (6)	238
Figure 260: Post survey result (7)	239
Figure 261: Dashboard of Zoho Inventory.....	240
Figure 262: Changes in order of sprint.....	241
Figure 263: Final Product Backlog	241
Figure 264: Lifecycles in FDD Model.....	242
Figure 265: Lifecycles of Spiral Model.....	244
Figure 266: Lifecycles involved in prototype model.....	246
Figure 267: Lifecycles of waterfall model	248
Figure 268:Logo of React JS.....	250
Figure 269: Logo of NodeJS	252
Figure 270: Logo of PostgreSQL (PostgreSQL, 2022)	253
Figure 271: System Architecture of the application	256
Figure 272: Screenshot of calling cron job	271
Figure 273: Screenshot of adding employee attendance.....	271
Figure 274: Screenshot of getting low stock notification.....	272
Figure 275: Screenshot of maintaining cron log.	272
Figure 276: Screenshot of global mail function	273
Figure 277: Cookie session setup	274
Figure 278: Screenshot of check role middleware.....	275

Figure 279: Screenshot of file upload middleware.....	276
Figure 280: Screenshot of global user notification	277
Figure 281: Screenshot of adding data to elastic search.....	278
Figure 282: Screenshot of function creating invoice PDF.....	278
Figure 283: Screenshot of branch maintained in Git.	279
Figure 284: Originality report.....	280

List of Tables

Table 1: Comparison of different similar project	14
Table 2: Justification for not choosing Waterfall model.	15
Table 3: Justification for not choosing Waterfall model.	15
Table 4: Justification for not choosing Waterfall model.	15
Table 5:Justification for not choosing Prototype model.	16
Table 6: Justification for not choosing Prototype model.	16
Table 7:Justification for not choosing Prototype model.	16
Table 8: Justification for not choosing Spiral model.	17
Table 9: Justification for not choosing Spiral model.	17
Table 10: Justification for not choosing Spiral model.	17
Table 11:Justification for not choosing Spiral model.	18
Table 12: Justification for not choosing Spiral model.	18
Table 13: Justification for not choosing Spiral model.	18
Table 14: Justification for choosing SCRUM.	20
Table 15:Justification for choosing SCRUM.	20
Table 16:Justification for choosing SCRUM.	20
Table 17:Justification for choosing SCRUM.	20
Table 18: Justification for choosing SCRUM. I	20
Table 19: Notable packages used in project.	25
Table 20: Login Page with invalid credentials	165
Table 21: Login Page with valid credentials	166
Table 22: Testing Login Form for SQL injection	168
Table 23: Testing new user registration with existing username.	169
Table 24: Testing new user registration with valid details.	170
Table 25: Testing to access protected route without login.....	171
Table 26: Testing to add product with duplicate id.	172
Table 27: Testing for adding a new product with valid details.	173
Table 28: Testing whether location updates based on the customer.	174
Table 29: Testing to add empty invoice.....	175
Table 30: Testing to change employee status.....	176

Table 31: Testing form validation for adding employees.....	177
Table 32: Testing for adding employee attendance.....	178
Table 33: Testing for adding duplicate entry for attendance.....	179
Table 34: Testing for adding new employee.....	180
Table 35:Testing to ensure data entry in elastic search database.....	181
Table 36: Testing for created search query.....	182
Table 37:Testing to get all notification for a specific user.....	183
Table 38: Table for testing whether customer receive email.....	184
Table 39: Testing to get supervisor dashboard data	185
Table 40: Testing to review record of sales.....	186
Table 41: Testing for generating report by supervisor	187
Table 42: Testing for downloading in PDF format.....	188
Table 43: Testing to downloading in CSV format.....	190
Table 44:Testing for adding data from CSV.....	191
Table 45: Testing for adding new packages.....	192
Table 46: Testing to generate QR Code for production package.....	194
Table 47: Testing for adding products by scanning package.....	195
Table 48: Testing for adding same package to the invoice.....	196
Table 49: Testing to access unauthorized API.....	197
Table 50: Testing authorized API proper access.....	199
Table 51: Testing whether users are directed to respective portal.....	201
Table 52: Testing to access route from direct URL	204
Table 53: Testing for viewing custom role for staff.....	205
Table 54: Testing Attendance Corn Job	207
Table 55: Testing Low stock Corn Job	208
Table 56: Testing Overdue alert Corn Job	210
Table 57: Roles and Responsibilities in FDD (digite, 2022)	243
Table 58: Roles and Responsibilities in Spiral Model (Sharma, 2020)	245
Table 59: Roles and Responsibilities in Prototype Model (Geeks for Geeks, 2022) ...	247
Table 60:Roles and Responsibilities in Waterfall Model (Shaikh, 2017).....	249

1. Introduction

Being in middle of two superpower countries which are known for producing huge amount of goods around the globe, Nepal could benefit from them. Nepal can boost the efficiency of its production operations, cut costs, and improve the quality of its goods. Additionally, the use of technology may boost the competitiveness of Nepalese manufacturers by allowing them to make items of greater quality and at a reduced cost, thereby offering new prospects for exports and enhancing the country's economic development.

Management software has become a crucial tool for firms of all sizes and sectors in today's digital world. It assists companies in increasing efficiency, reducing mistakes, and making better decisions. (Wan, et al., 2016). Country currently placed in 8th position for developing manufacturing software, it still lacks behind for implementing content management system in small scale industry. (Shrestha, 2021)

Based on the above finding, an application for improving management inside an industry of small scale was chosen for project. Using this application, a small-scale industry can manage their day-to-day activities, analyze their data, and manage role-based activities inside this application. The application will be a web-based solution making it more accessible without any operating system barriers.

1.1. Project Overview

The project is named as **Udhyog**, a translation to industry in Nepali language. This project will create a web-based application following **SCRUM** as its development methodology. The project will be divided into different sprint and will be implemented following principles of SCRUM methodology.

The built web-based application is a major contribution towards the digitalization and of industries. It is created using Node.js as the backend and React.js as the frontend, making it a scalable, resilient, and user-friendly application. The REST API is used to communicate data between the two components, while the PostgreSQL database is utilized to hold all the essential information needed to administer an industry. This application could simplify industrial operations and increase overall efficiency.

1.2. Current Scenario

1.2.1. In context of Nepal

According to current data, Nepal recorded a trade deficit of more than 11000 million NRP. (Nepal Rastriya Bank, 2023). With population highly depending on imported products, industry in Nepal is not able to reach its potential. It requires to boost different sector to boost its economy and gain the goal they have set. It has been proven that use of proper technology in business helps business achieve big achievement. However, this industry has always been left behind when it comes to technology. (Shrestha, 2021)

With the use of the advance technology available today in the market, the industry around Nepal cannot compete with the international market which are way ahead of us. The owner of the industry is relaying inaccurate, redundant data. With no proper management system, the task must be done by individual personal causing extra use of human resources. Also, the industry in Nepal heavily relays on the file bases data collection. In case of any disaster, the important information can be destroyed. With such, use of digital management system could help industry in Nepal to grow exponentially. (Shrestha, 2021)

1.3. Problem Domain

1.3.1. Problem Domain Listed from research.

- Traditional way of storing data being unreliable, unmaintainable, and unsecure (Shrestha, 2021)
- Waste of valuable human resource for maintaining the traditional approach of stock management.
- 50% of businesses around the globe not using big data for growing their business. (Ku, 2021)
- Industry digitalizing the data storing approach also not converting data to useful information.
- Business analytics being expensive for small-scaled industry.

1.3.2. Problem Domain Listed from survey.

- Expensive software
- Non-relatable feature in context of Nepal.
- Conversion of date being hectic for user.
- Different human error causing loss in business.

1.4. Project Scope

From the above scenario, a conclusion can be conducted that small-based industry are facing huge problem due to their traditional approach of storing the data. As such below are some solutions that might help industry solve the problem:

- Creating an application for storing the collection in digital format.
- The application will just have an admin panel where the user can interact with the application's various functionalities.
- The application is a hybrid of various management systems merged into a single application.
- The program would be able to digitally record all the company's needed activities and create useful data.
- To be useful by the owner, the application's user interface (UI) must be simple to use.
- The application will also be able to provide different reports on the daily activities performed by the industry to help them understand the data in meaningful information.

1.5. Aims and Objectives

1.5.1. Aim of the Project

The project's primary goal is to create a dependable, user-friendly, and successful application for the customer that will not only solve the problem described above but will also allow the client to grow new paths in company using the application.

1.5.2. Objectives of the Project

The following objectives must be met to attain the goals:

- Gather information about similar goods that offer functionality to solve analogous difficulties in the order business.
- To keep the client's vital information, secure by maintaining a secure system.
- To obtain real experience in developing software while adhering to the entire software development lifecycle.
- To assist clients in growing their businesses by resolving all the problems listed.
- Correctly understanding the client's requirements and delivering software in accordance with them.
- Use suitable development process to complete projects on schedule.

1.6. Structure of the report

1.6.1. Introduction

The introduction of this report consists of short description about the project, why the project is a necessity in current context, aims and objective of the project and major problem domain. The section provides a narrow understanding on the project and the structure it will follows.

1.6.2. Background

This portion of reports gives clarification about the intended user type for the project. The part will also provide a short analysis on comparable project presently being utilized in the market. Various characteristics supplied by the related projects will be compared with this project. The part will also be included with brief information on tools and technologies utilized to finish the project.

1.6.3. Development

This section documents the methodology used in the project, the processes undertaken in different time frames, and the tasks carried out to build the application. It includes development coding, requirement analysis, reviews of the development process, and steps taken to improve the response received from feedback. Automatic test cases generated sprint wise will also be added to the section.

1.6.4. Testing and analysis

This section documents the testing of the software development project, including different test cases and scenarios to verify the application's quality. The results of the tests will be analyzed, and necessary changes to the development will be implemented if required. The section will only include manual testing conducted during and after completing the project. Finally, a critical analysis of the development process will be conducted in this section.

1.6.5. Conclusion

This section provides an analysis on the developed application. The analysis includes topic like limitation of the project, positive impact of the project on user, future features that could further help user improve their experience on the application. The section will also discuss about the challenges faced while completing the project. Finally, the section will also provide steps carried to solve limitation and challenges.

2. Background

2.1. Targeted User Type

As aimed, the project is targeting any small, scaled industry wanting to integrate management system. The application would be a huge game changer for industry currently working on file-based management system. As such, any industry facing problem through file-based management, industry looking for easier solution for management are the targeted user for the project.

2.2. End User

2.2.1. Client Information

For the project, I have reached out to a socks factory New Bhavuk Hosiery located in Kathmandu. The company was established in 2059 B.S. During my visit to the factory, I found the company to be completed based manual recording. The company have been providing fashionable socks through the market from early days. According to the founder of the company, they have been producing around 72,00,00 pairs sock every year. The company is running with around 15-20 employee. However, with company planning to expand in near future, file-based recording could be problematic. Hence, a proposal for the developing an application to manage daily workflow of the factory was send to the owner of the company.

2.3. Understanding the solution

Udhyog is an alternative to file-based data storing which seems to be very popular in Nepal. With numerous disadvantages of the file-based storage system, this project helps user move to the face of digitalization and grasp benefits from it which will provide positive feedback in their organization.

2.3.1. Solution for Client.

This project prioritizes meeting the client's requirements to overcome inaccuracies and redundancies in the client's organization's data. The project will provide detailed insights into the organization's workings, serving as an additional asset. The project will develop a web application using React, Nodejs, and PostgreSQL, including all required features, with any additional client requirements considered for future work after project completion.

2.4. Review of similar project

2.4.1. Swastik Business Accounting Software

Swastik Business Accounting Software is a product developed by Hitech Solutions & Services Pvt Ltd. The system has been a popular solution for accounting which also includes feature of manufacturing inventory. The system allows its user a window only solution which to create billing of any kind of products and services. The software does not contain role-based system making it vulnerable. The software is also local solution which does not allow the data sharing outside (HiTech Solution & Services Pvt Ltd, 2022).

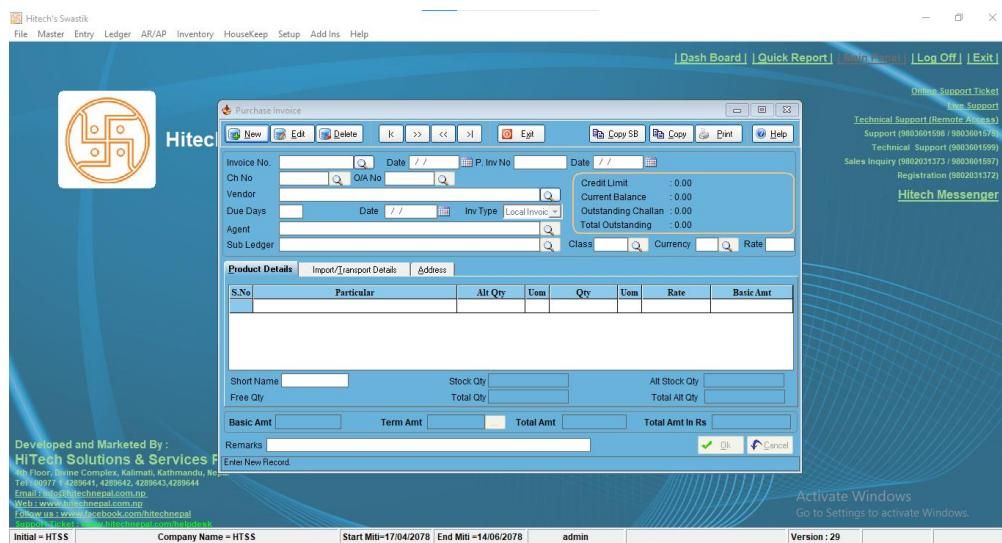


Figure 1: Screenshot of Swastik App

Source: (HiTech Solution & Services Pvt Ltd, 2022)

Being most popular billing software in Nepal, above are some features that inspired the project:

- Inventory Management system
- Credit based billing.

2.4.2. Vyapar App

Vyapar is a FREE Business Accounting Software designed for Indian small businesses to handle invoicing, inventory, accounting, and other tasks. The idea is to make a businessman's daily routine less taxing and to allow him to focus on building his firm rather than paperwork. The app is a popular solution with huge number of downloads. However, the app lacks the features for management of the manufacturing production. (Vyapar App Pvt Ltd, 2022)



Figure 2: Vyapar App

Source: (Vyapar App Pvt Ltd, 2022)

With a good eye-catching design to advance report generation, the project has inspired the project with different features. Some of the inspired features are:

- Good UI
- Advance Report Generation
- Payment Overdue Alert

2.4.3. Tally

Tally is a commercial accounting and inventory management software tool. It is intended to assist businesses in keeping track of their financial activities such as sales, purchases, receipts, and payments. Tally can create financial reports including profit and loss statements and balance sheets, as well as manage inventories and track the flow of items. It is frequently utilized for small and medium-sized enterprises in India and other nations. (Tally, 2022)

Some of the features that inspired the project are:

- Payroll System
- Professional Invoice System

Description on Zoho Inventory is included in [appendix section](#)

2.5. Comparison Between System

A complete difference between all the above listed project and my project is listed in a tabular format below:

S. N	Features	Udhyog	Swastik	Vyapar	Tally	Megav entory	Zoho
1	Login Authentication	✓	✓	✓	✓	✓	✓
2	Register	✓	✓	✓	✓	✓	✓
3	Role Based System	✓	✗	✗	✗	✓	✗
4	Employee Management	✓	✗	✗	✗	✗	✗
5	Warehouse Based Product	✗	✗	✓	✓	✓	✓
6	Company Expense Report	✓	✗	✓	✓	✓	✗
7	Custom Role Management	✓	✗	✗	✗	✗	✗
8	QR code-based billing	✓	✗	✗	✗	✓	✓
9	Inventory Management	✓	✓	✓	✓	✓	✓
10	Payment Overdue Alert	✓	✗	✓	✗	✓	✓
11	Low Stock Alert	✓	✓	✓	✓	✓	✓
12	Report Generation	✓	✓	✓	✓	✓	✓

Table 1: Comparison of different similar project

3. Development

3.1. Considered Methodology

3.1.1. Waterfall Model

Waterfall Methodology can be considered as a classical methodology popular in early days. The model with its simplicity was the reason for its popularity. Likewise, with waterfall model it follows the steps the software development lifecycle step by step. With waterfall model the phases of the SDLC never overlap. Also, the model does not have any turning point i.e., once a step is completed the development team cannot move back to phases. (Sharma, 2021)

Brief description on waterfall model is included in [appendix section](#).

3.1.1.1. Reason for not choosing Waterfall Model

Scenario	Being academic project, small development progress needs to be submitted to supervisor every week
Justification	With this model, every step of SDLC can only be followed one after another.

Table 2: Justification for not choosing Waterfall model.

Scenario	With client involved, the requirement of the project might change in the future.
Justification	Following step-by-step process, planning need to be carried out early of development which means future changes are not accepted.

Table 3: Justification for not choosing Waterfall model.

Scenario	With short period of time for the project, test of the product might not have sufficient time.
Justification	With this model, testing is only carried out at the end of the completion of the development. This means bugs are not caught until it's late.

Table 4: Justification for not choosing Waterfall model.

3.1.2. Prototype Methodology

The Prototype model is a client-centric methodology that relies on customer satisfaction. Development teams build a prototype for the client's requirements, and the project is considered complete only when the client is satisfied with the prototype. This model is useful when the client is uncertain about the project requirements, and it allows for the selection of different features to build the desired product. The development team builds a prototype that is evaluated by the client, and the cycle iterates until the client is satisfied with the final product. (Geeks for Geeks, 2022)

Brief description on prototype methodology is included in [appendix section](#).

3.1.2.1. Reason not choosing Prototype Model

Scenario	The entire success of the project is dependent on the feedback from the client.
Justification	With this model, the progress of the project is measured with feedback from client, meaning project might not reach its goal with deadline

Table 5: Justification for not choosing Prototype model.

Scenario	Unable to create proper scheduling for the project.
Justification	Following this methodology, the project will not be able to schedule properly as we might have to create multiple prototypes.

Table 6: Justification for not choosing Prototype model.

Scenario	Being a single person project, I lack resources.
Justification	Following this methodology, we require huge amount of the resources as prototype are required to be developed rapidly.

Table 7: Justification for not choosing Prototype model.

3.1.3. Spiral Model

The Spiral Model is an iterative software development methodology that focuses on reducing project risks. It operates similarly to the Waterfall model and Prototype model but runs in a spiral iteration (Software Testing Help, 2022). The development team builds a small-scale version of the product, assesses the risks involved, and documents the plans to handle them before deciding whether to proceed to the next step. This cycle continues until the final product is ready. The Spiral Model is a complex process that could continue indefinitely (Sharma, 2020).

Brief description about spiral model is included in [appendix section](#).

3.1.3.1. Reason not choosing Spiral Model

Scenario	Being a single handle project, we lack resources.
Justification	Following this methodology, much of the time and resources are wasted on determining the risk on the project.

Table 8: Justification for not choosing Spiral model.

Scenario	Unable to create proper scheduling for the project.
Justification	Following this methodology, the project might go on infinite spiral determining the risk.

Table 9: Justification for not choosing Spiral model.

Scenario	Being familiar with the work around of the requirement needed, the project doesn't seem to be risky
Justification	The methodology is not suitable for the project with low risks.

Table 10: Justification for not choosing Spiral model.

3.1.4. Feature Driven Development (FDD)

Feature Driven Development is an agile framework, solely based on deliverable of the features included in the project. The features included in the FDD are not features of the project but features related to user stories. FDD is also known to fill the gap between traditional methodology and popular methodology like SCRUM.

Brief discussion about FDD is included in [appendix section](#).

3.1.4.1. Reason not choosing FDD Model

Scenario	This project is being developed by single person and must fulfill the role and responsibility of all required person in the project.
Justification	The methodology is not suitable for the project with small team as different person must perform different role to accomplished success.

Table 11: Justification for not choosing Spiral model.

Scenario	This project requires to build documentation of every step taken while build the project.
Justification	Using the methodology, it focuses primarily on delivering the features rather than documentation

Table 12: Justification for not choosing Spiral model.

Scenario	This project is the first project I have worked as project manager.
Justification	The methodology succession requires an expert on the project which this project certainly lacks.

Table 13: Justification for not choosing Spiral model.

3.2. Selected Methodology

3.2.1. About Scrum

Scrum is an agile technique that emphasizes cross-functional teamwork, responsibility, and iteration to develop, deliver, and support its clients. The scrum technique allows for the necessary adjustments to be made to the project. The project will also be able to progress within the time range specified. The complex project is broken into small chunks using the scrum approach, which is also known as sprint. (Mountain Goat Software, 2022) The sprint is the backbone to the scrum. In the early planning phase of the project, the large project will be divided in the smaller part which will be completed in the given timeframe. Likewise, with following scrum, the project will not have any project leader. (Mountain Goat Software, 2022)

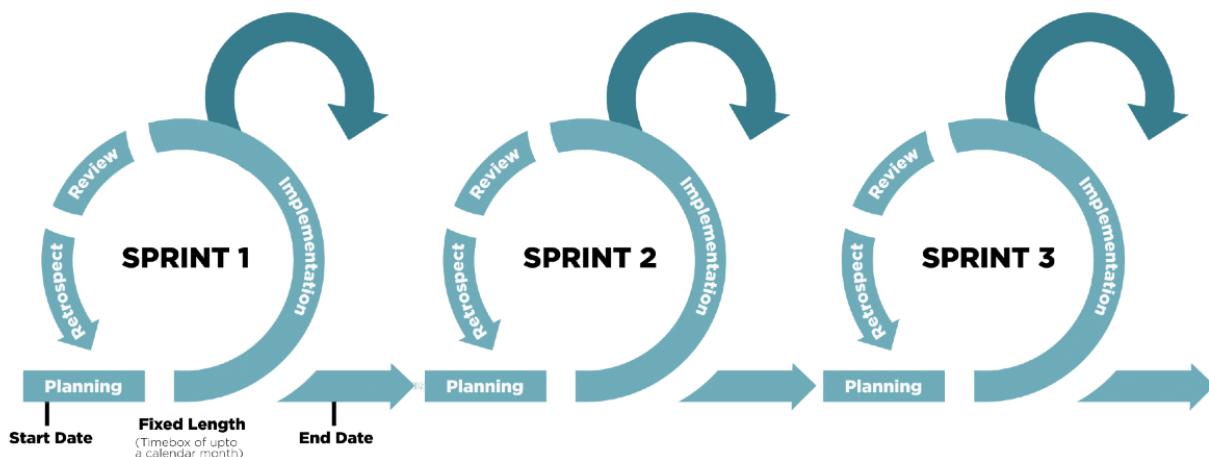


Figure 3: Sprint Cycle with its sub process

Source: (Henderson, 2022)

3.2.2. Reason choosing SCRUM.

The following were the justification for choosing SCRUM methodology for the project.

Scenario	With client involvement, changes in the project might occur after planning is completed
Justification	With Scrum, changes are happily accepted. (Mountain Goat Software, 2022)

Table 14: Justification for choosing SCRUM.

Scenario	With client involvement, changes in the project might occur after planning is completed
Justification	With Scrum, changes are happily accepted. (Mountain Goat Software, 2022)

Table 15: Justification for choosing SCRUM.

Scenario	Requirement to show progress of the project with supervisor.
Justification	The methodology allows the planning and execution on iteration allowing project to start from early phase

Table 16: Justification for choosing SCRUM.

Scenario	First project as project manager
Justification	The methodology allows continuous improvement as it overlook the result of each sprint.

Table 17: Justification for choosing SCRUM.

Scenario	The project is more focused on academic side with strict deadline
Justification	The methodology is justified as accurate methodology by maintain the product backlog and scrum artifacts.

Table 18: Justification for choosing SCRUM. I

3.3. Phases of Methodology and its implementation

3.3.1. Initiation

Being a first phase of the project, the phases focus on the vision of the project. The phase requires management to set a goal for the project. The phase prioritized on gathering information from client and creating documentation using it. (Mountain Goat Software, 2022) Task required in initiation are:

- Defining aims and objective
- Understanding competition in market
- Creating product backlog using user stories

3.3.2. Planning and Estimation

Usually, planning in the scrum runs till the end of the project. Developing sprint cycle to overrun the project smoothly is a crucial role in planning. After breaking down the sprint the planning and estimation works on iterative cycle. (Indeed Editorial Team, 2021) Task required in planning and estimation are:

- Breaking Product Backlog to Sprint
- Determine goal of each Sprint
- Planning execution of Sprint

3.3.3. Implementation

During the phase, the sprint team works on the planned project. Creating the product and performing action on the product helps team reach a step closer to the goal. Updating product backlog as progress is an important task of this phase. (Indeed Editorial Team, 2021) Other tasks involved in this phase are:

- Writing code and developing UI
- Performing Daily Scrum meeting
- Creating prototype using wireframe developed.
- Testing on the code

3.3.4. Reviewing

As Scrum rely heavily on client input, it has a phase to get review for client after execution of each Sprint. The meeting is also referred as review meeting. In this meeting, the overall execution of the Sprint is discussed and brainstorm ideas to improve the work completed. (Indeed Editorial Team, 2021). Task required in reviewing are:

- Make changes in product backlog.
- Getting feedback from the client and supervisor

3.3.5. Releasing

The last step of Scrum requires to deliver the final product developed to the client. Reviewing all the sprint and evaluating the performance in the Sprint is also carried out.

- Deploying the final product
- Testing in the live server

3.4. Survey Results

3.4.1. Highlights from Pre-Survey

- The survey was conducted manually with different employees, management and owner of small-scaled industry based on Nepal.
- 6 responses were received from the survey where problem faced by the industry in Nepal were asked.
- Many industries in Nepal are still using file-based system to store daily activities. Most industry using other application were also found not to be satisfied.
- Complex & expensive system being cause for industry to move to any application.
- Nepali Date Format was taken as highest priority from the survey.

Survey Result are included in [appendix section](#).

3.4.2. Highlights from Post-Survey

- The survey was conducted online using Google Forms, where different individual was given certain time to use the application.
- 11 responses were received as part of this survey.
- Most people in survey finds the application to be useful for industry in Nepal.
- More than 63% of the people in survey believe mobile application of this project would be an additional feature.
- Positive responses were received for the overall experience with the application.

Survey Result are included in [appendix section](#).

3.5. Requirement Analysis

The process of collecting, comprehending, and recording the client's specifications for a software product is known as requirement analysis. It entails determining the client's wants, hopes, and goals and turning them into functional and non-functional specifications that will direct the development process. The major features requirements for the project are:

3.5.1. Software Requirements

- React JS – Frontend JS Library ([Appendix Section](#))
- HTML / CSS – Frontend structure. ([Appendix Section](#))
- Tailwind CSS ([Appendix Section](#))
- Node JS – Backend JS Library. ([Appendix Section](#))
- PostgreSQL – Database. ([Appendix Section](#))
- Elasticsearch – Search Database ([Appendix Section](#))
- Jira – Project Management
- Git/GitHub – Version Control / Remote code storage
- Balsamiq – Wireframes Designs
- Figma – Mockup Designs
- Word – Creating documents.
- VS Code – Text Editor
- Postico – Database Client GUI
- Team Gantt – Gantt Chart Development
- Draw.io – Diagram development

3.5.1.1. Notable Packages used.

Frontend	Backend
<pre> "@sbmdkl/nepali-date-converter": "^1.3.3", "@sbmdkl/nepali-datepicker-reactis": "^-1.2.1", "@testing-library/jest-dom": "^5.16.5", "@testing-library/react": "^13.4.0", "@testing-library/user-event": "^13.5.0", "axios": "^1.1.2", "chart.js": "^4.2.1", "date-fns": "^2.29.3", "faker": "^6.6.6", "html2canvas": "^1.4.1", "jspdf": "^2.5.1", "leaflet": "^1.9.3", "lodash": "^4.17.21", "papaparse": "^5.4.0", "react": "^18.0.0", "react-chartjs-2": "^5.2.0", "react-csv": "^2.2.2", "react-data-table-component": "^7.5.3", "react-dom": "^18.2.0", "react-hook-form": "^7.37.0", "react-leaflet": "^4.2.1", "react-gr-code": "^2.0.11", "react-qrcode-reader": "^1.1.3", "react-router-dom": "^6.4.1", "react-scripts": "5.0.1", "react-select": "^5.4.0", "react-table": "^7.8.0", "react-toastify": "^9.0.8", "react-use-file-upload": "^0.9.5", "web-vitals": "^2.1.4" </pre>	<pre> "@elastic/elasticsearch": "^8.5.0", "@sbmdkl/nepali-date-converter": "^1.3.3", "bcrypt": "^5.1.0", "body-parser": "^1.20.0", "cookie-parser": "^1.4.6", "cors": "^2.8.5", "csv-parser": "^3.0.0", "date-fns": "^2.29.3", "dotenv": "^16.0.3", "express": "^4.18.1", "jsonwebtoken": "^8.5.1", "luxon": "^3.3.0", "multer": "^1.4.5-lts.1", "nodemailer": "^6.9.0", "nodemailer-express-handlebars": "^6.0.0", "nodemon": "^2.0.20", "pdfkit": "^0.13.0", "pdfkit-table": "^0.1.99", "pg": "^8.8.0", "pg-format": "^1.0.4", "pg-hstore": "^2.3.4", "sequelize": "^6.25.0" </pre>

Table 19: Notable packages used in project.

3.6. Design

3.6.1. Application Logo



Figure 4: Application Logo

3.6.2. System Design

3.6.2.1. Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) can summarize the system's users and their interactions with the system. To construct one, you will employ a collection of specialized symbols and connectors. An efficient use case diagram can assist your team in discussing and illustrating.

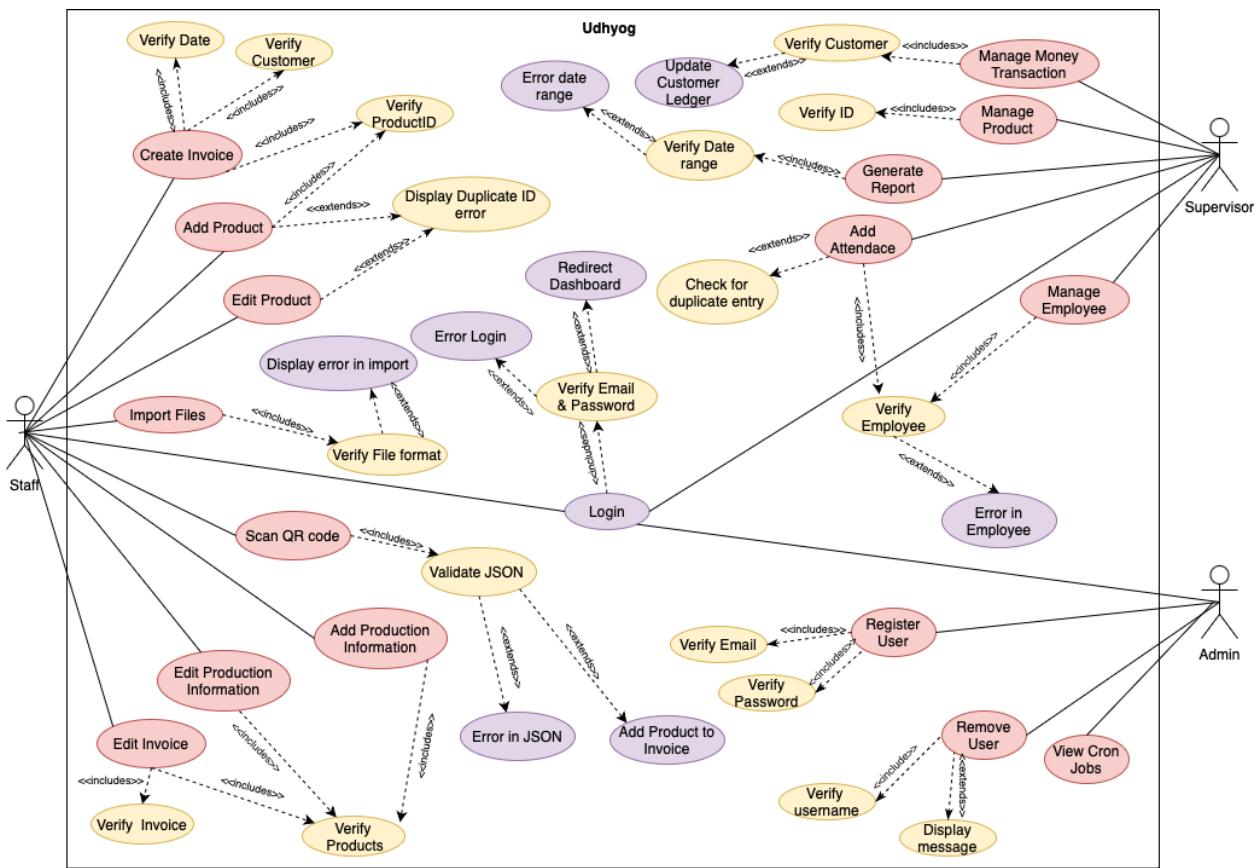


Figure 5: Use case diagram

Use Case Diagram iteration can be found in [appendix section](#).

3.6.2.2. ER Diagram

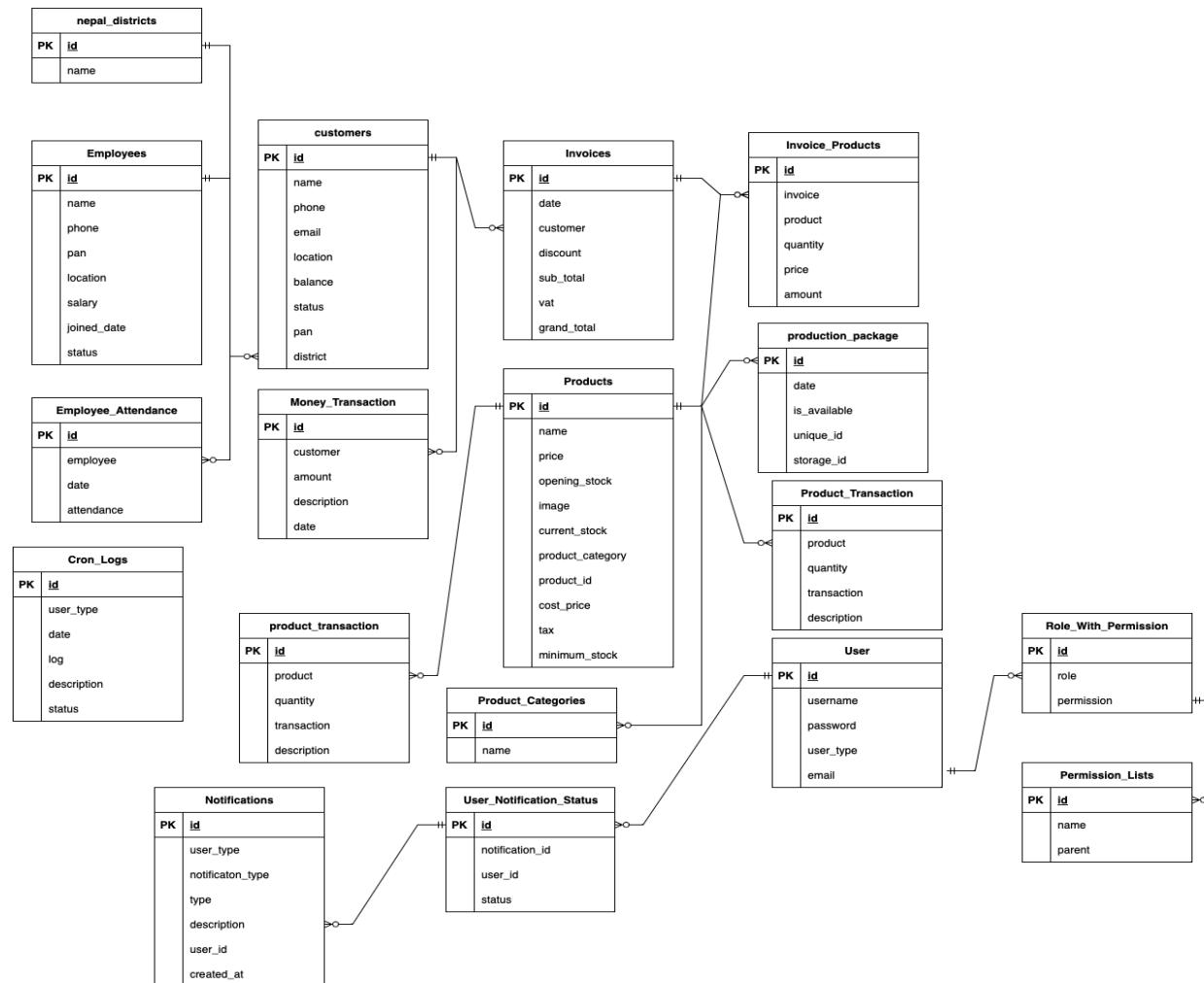


Figure 6: Final ER Diagram for project

ERD Iteration can be found in [appendix section](#).

3.6.2.3. Gantt Chart

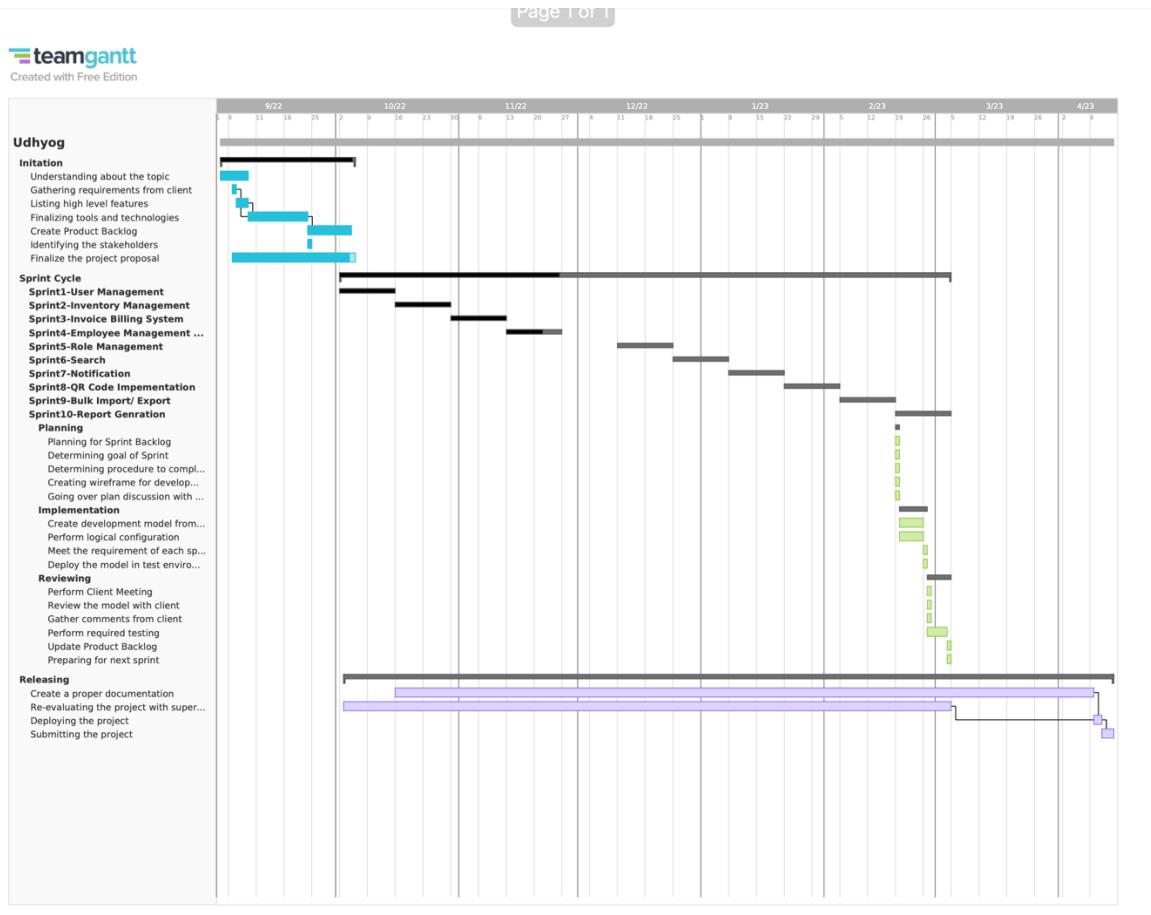


Figure 7: Final Gantt Chart

Gantt Chart iteration can be found in [appendix section](#).

3.6.2.4. Activity Diagram

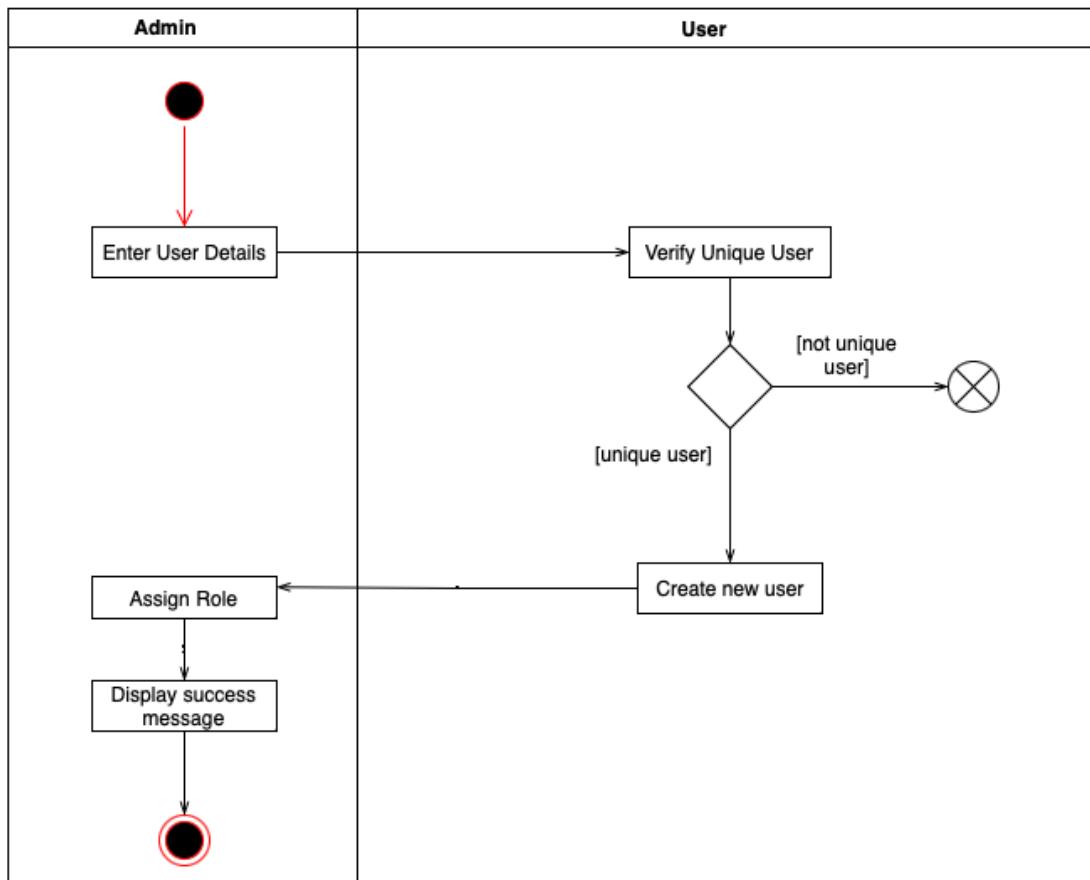


Figure 8: Activity Diagram for registering new user.

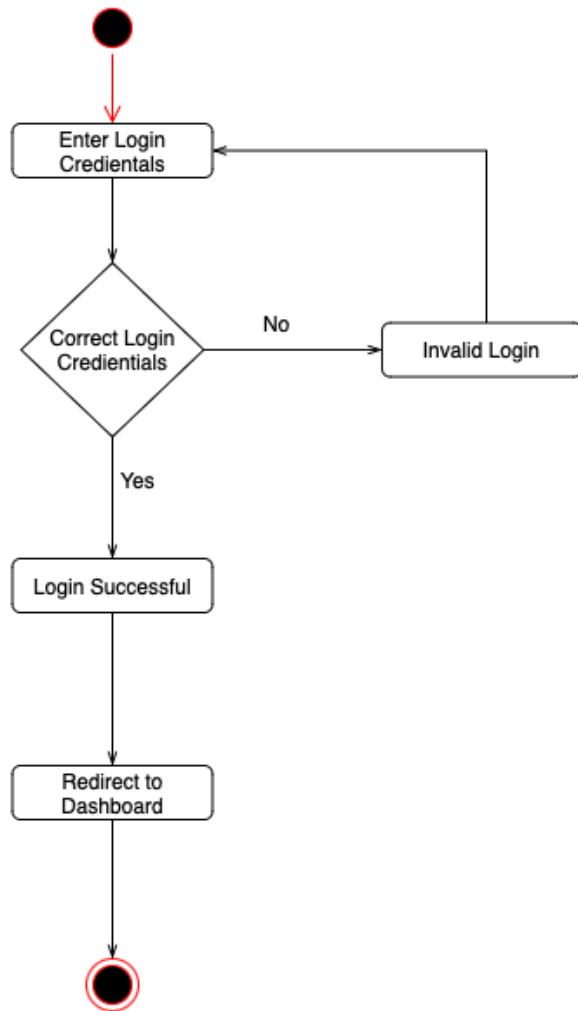


Figure 9: Activity diagram for accessing dashboard.

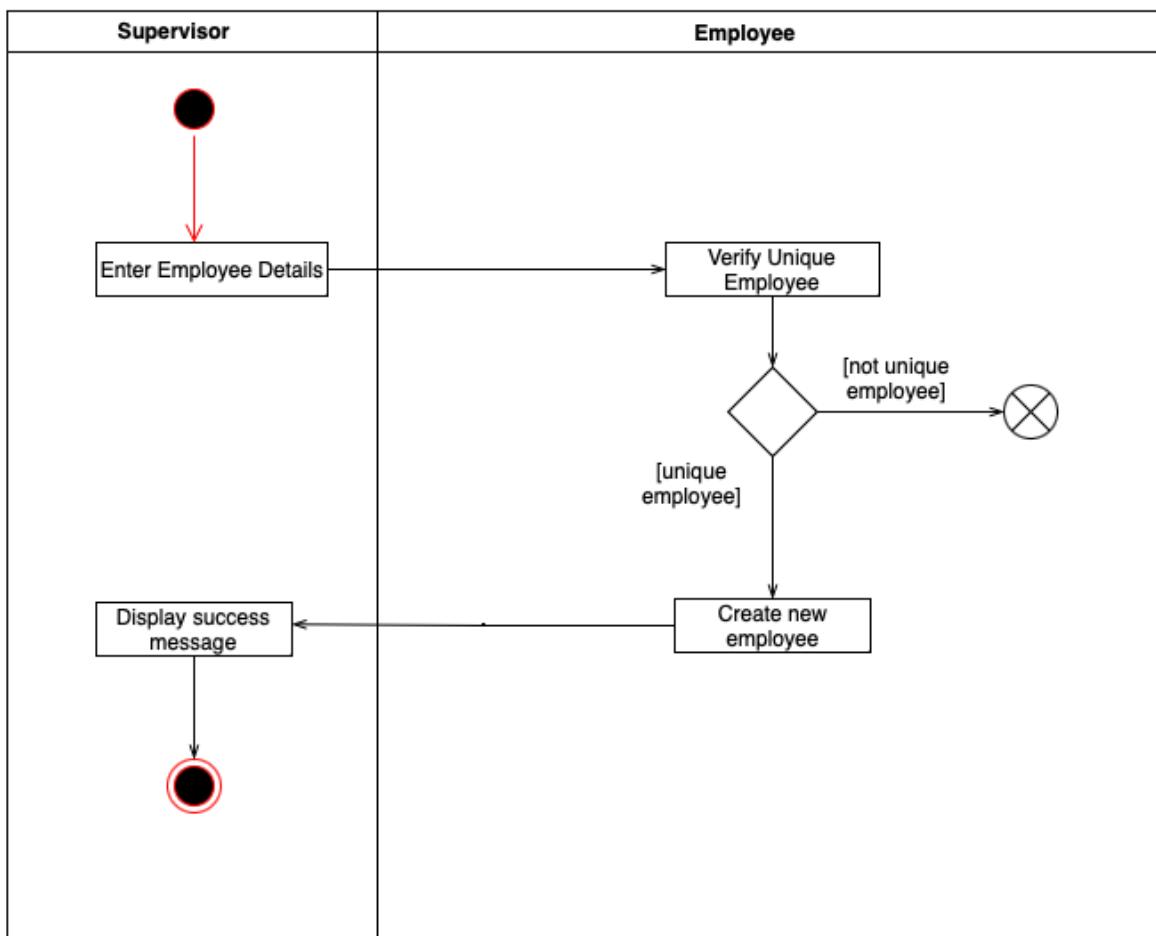


Figure 10: Activity diagram for adding new employee.

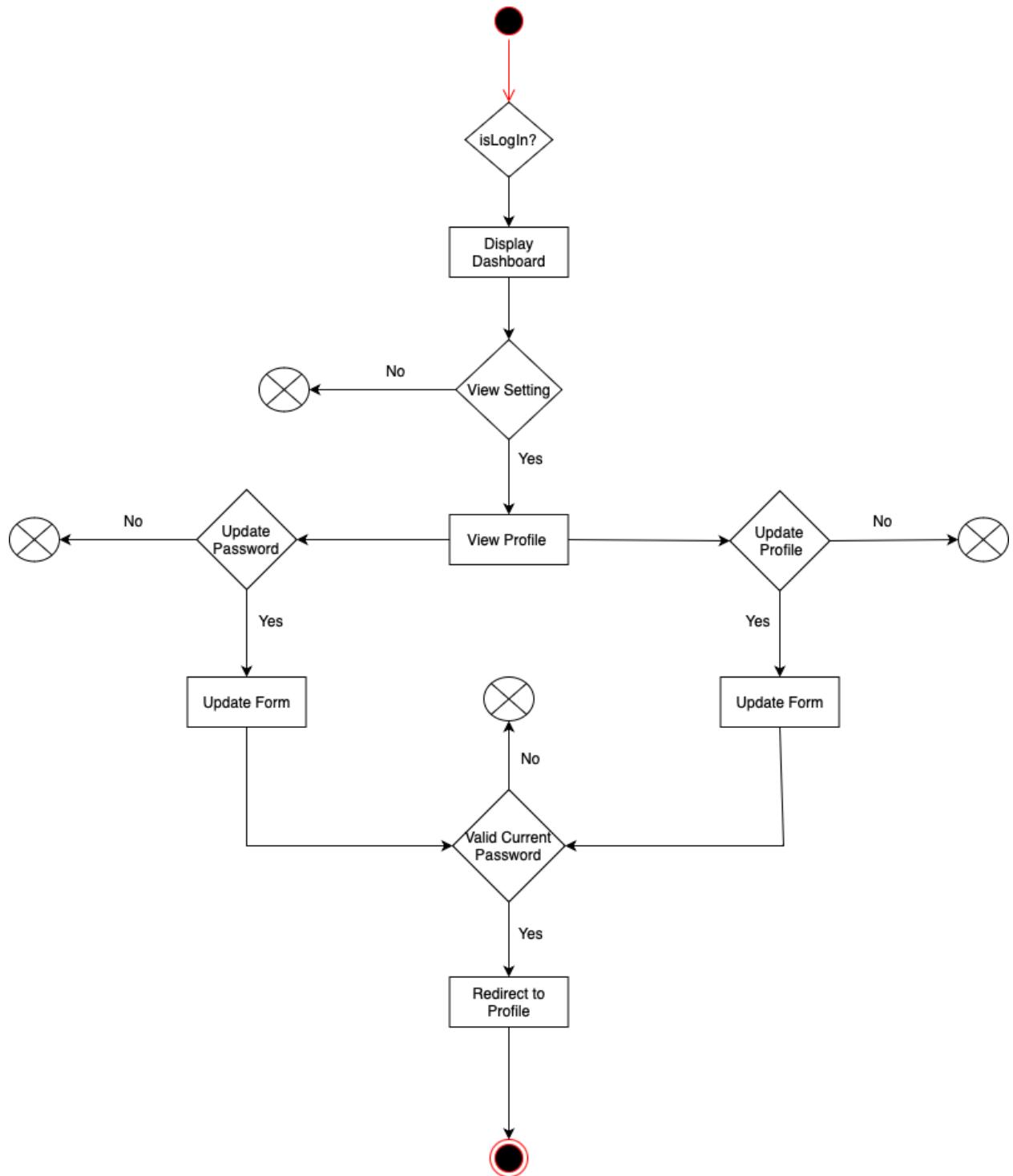


Figure 11: Activity Diagram for user setting

3.6.2.5. DFD Diagram

- Context Level Diagram

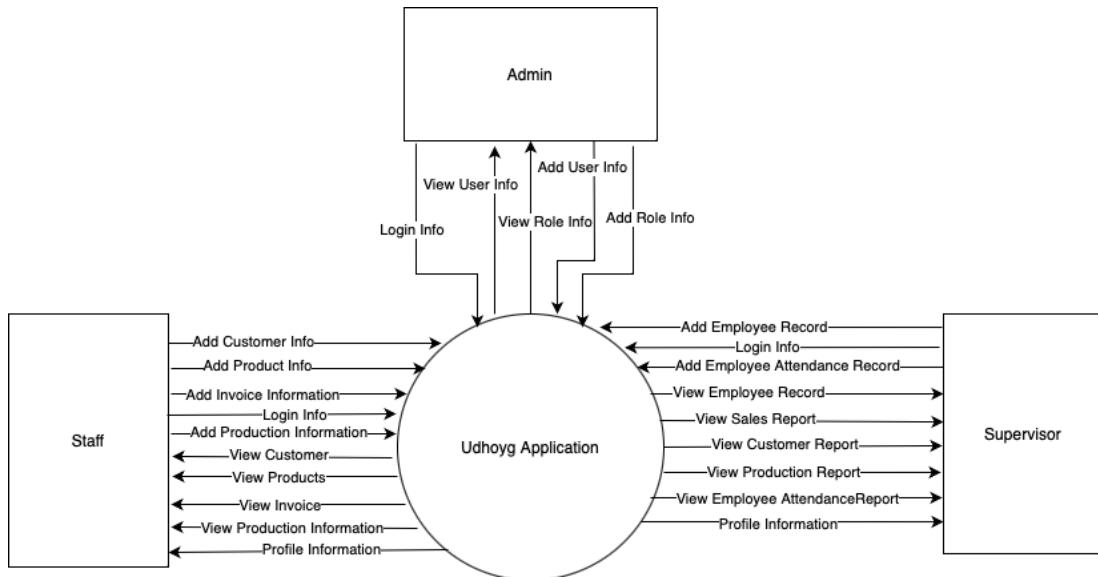


Figure 12: Context Level Diagram for Udhayog

- DFD Level 1

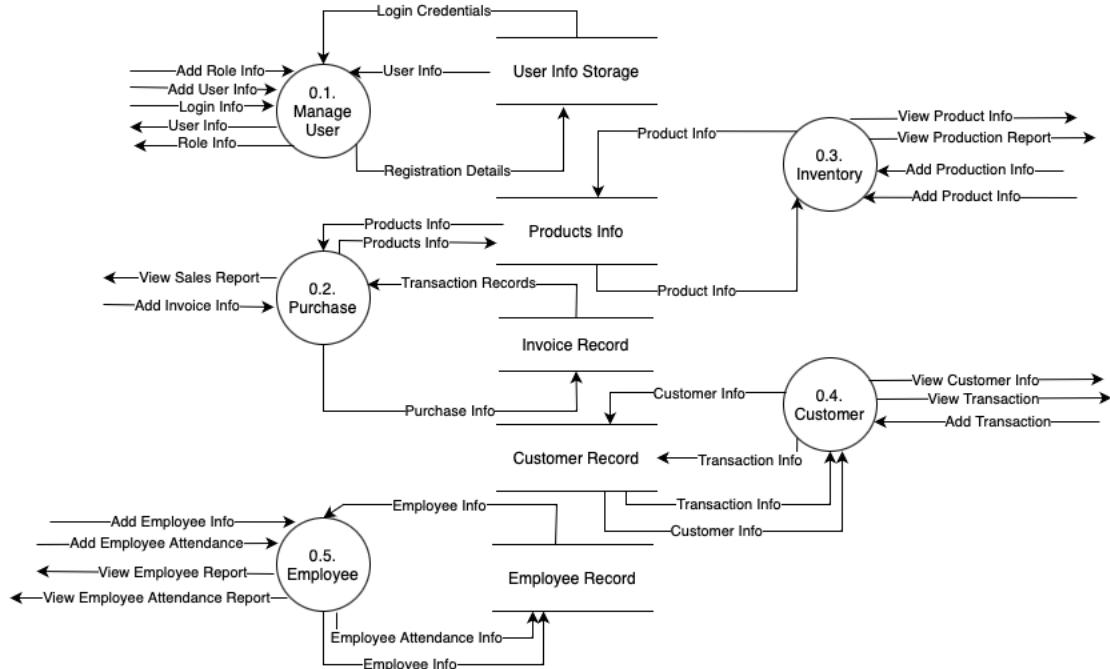


Figure 13: Level 1 DFD for Udhayog

3.6.3. Feature Design

3.6.3.1. Register User

a) Data Flow Diagram

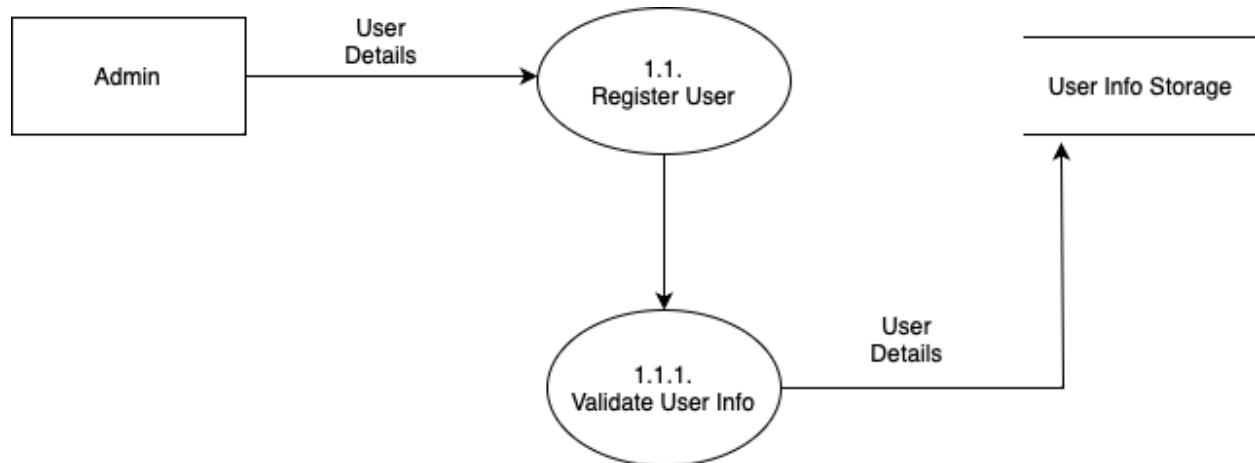


Figure 14: DFD Level 2 for registering new user.

b) Communication Diagram

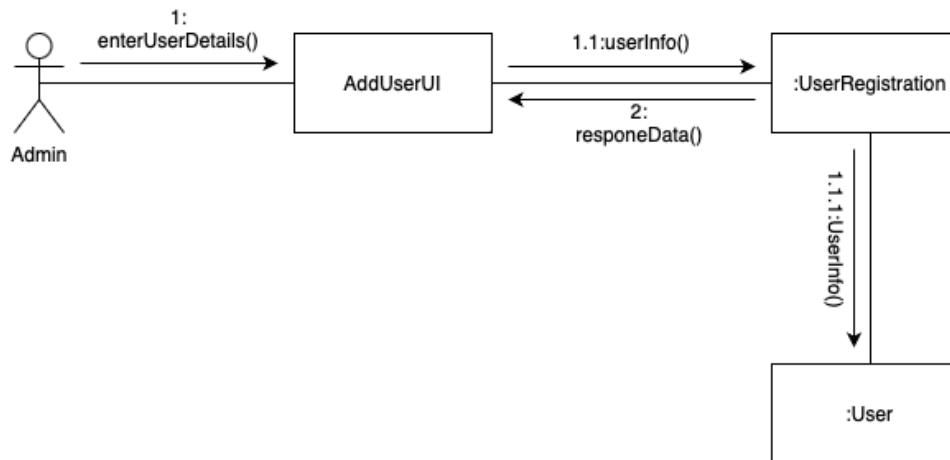


Figure 15: Collaboration Diagram for registration

c) Sequence Diagram

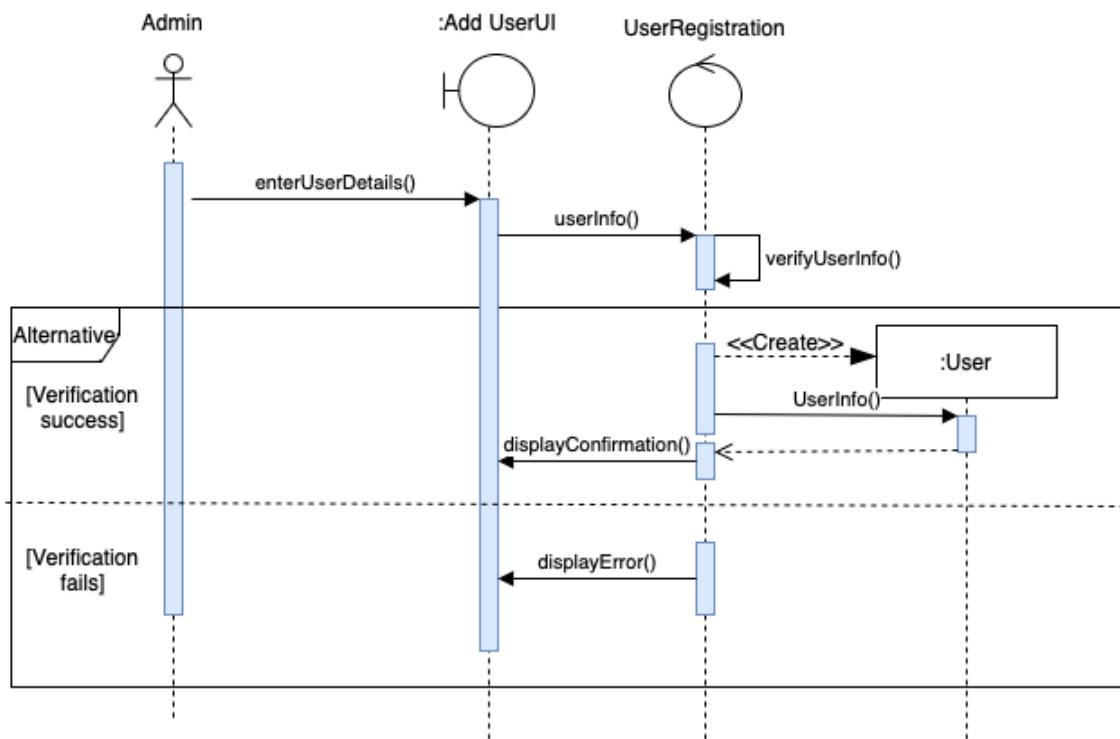


Figure 16: Sequence Diagram for registering new user.

3.6.3.2. Login User

a) Data Flow Diagram

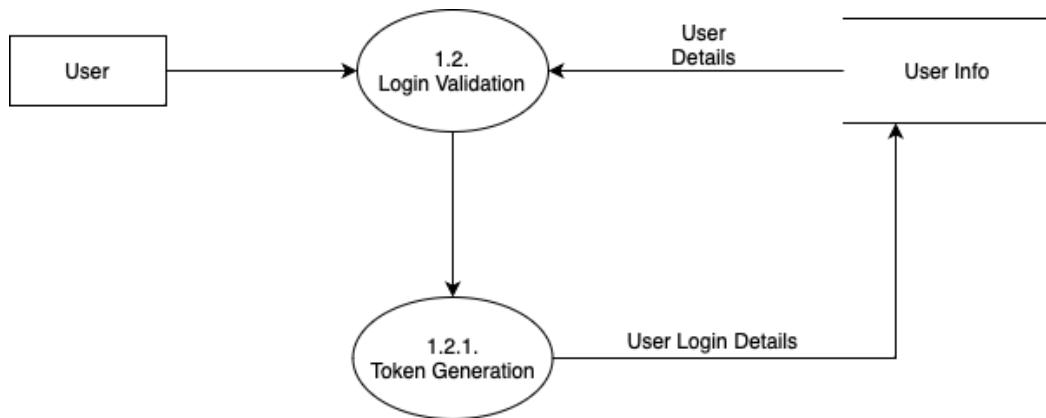


Figure 17: DFD Level 2 for login user

b) Communication Diagram

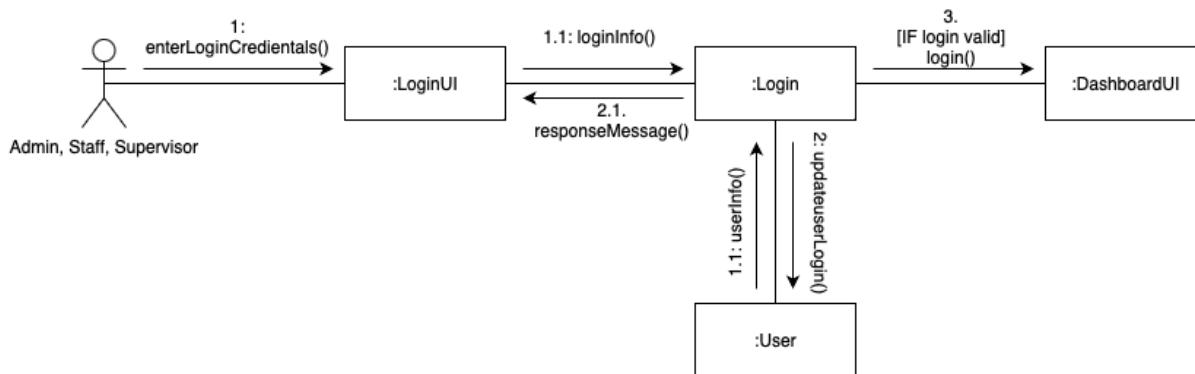


Figure 18: Communication Diagram for Login

c) Sequence Diagram

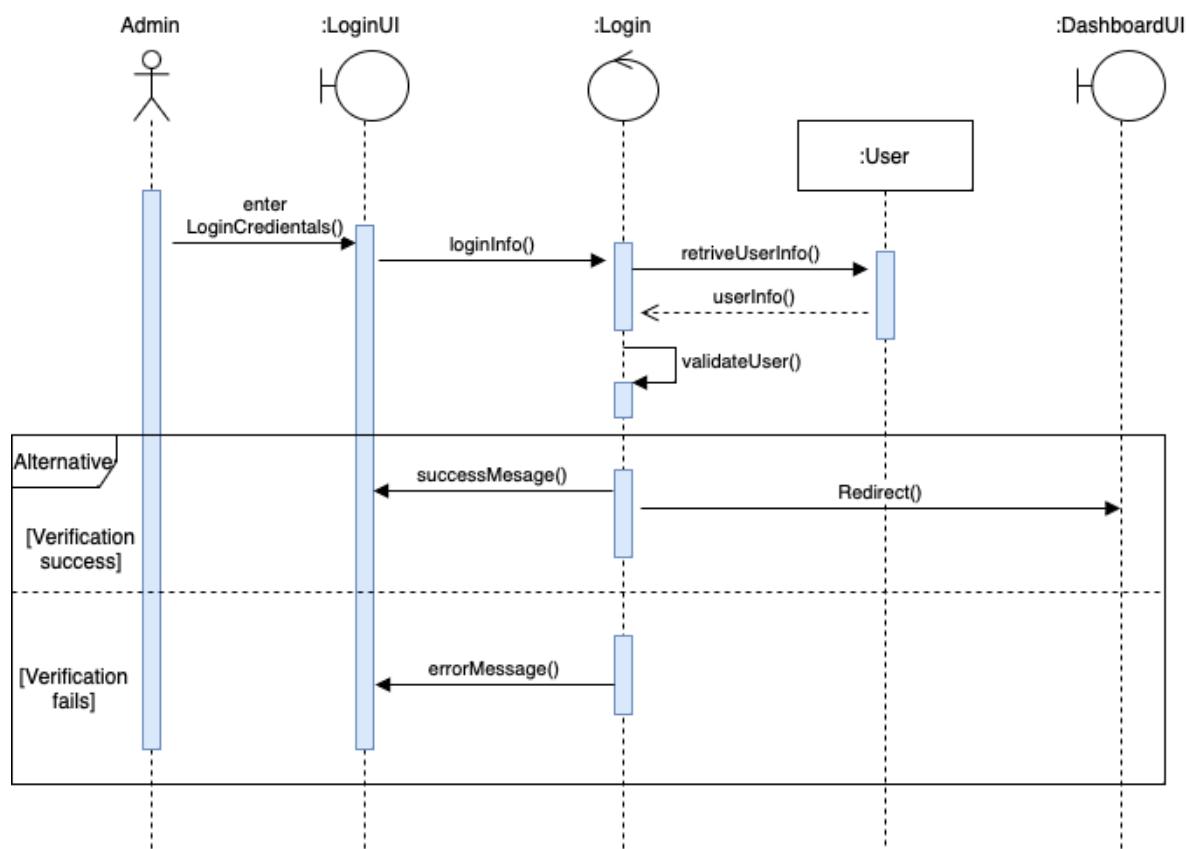


Figure 19: Sequence Diagram for Login

3.6.3.3. Generate Invoices

a) Data Flow Diagram

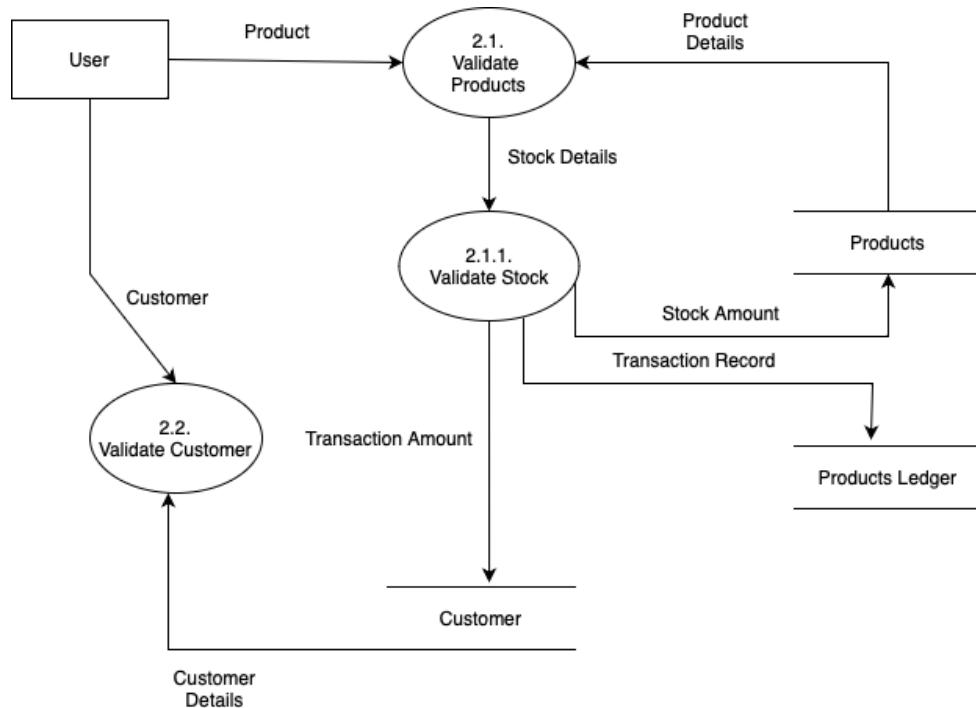


Figure 20: DFD Level 2 for generating invoices.

b) Communication Diagram

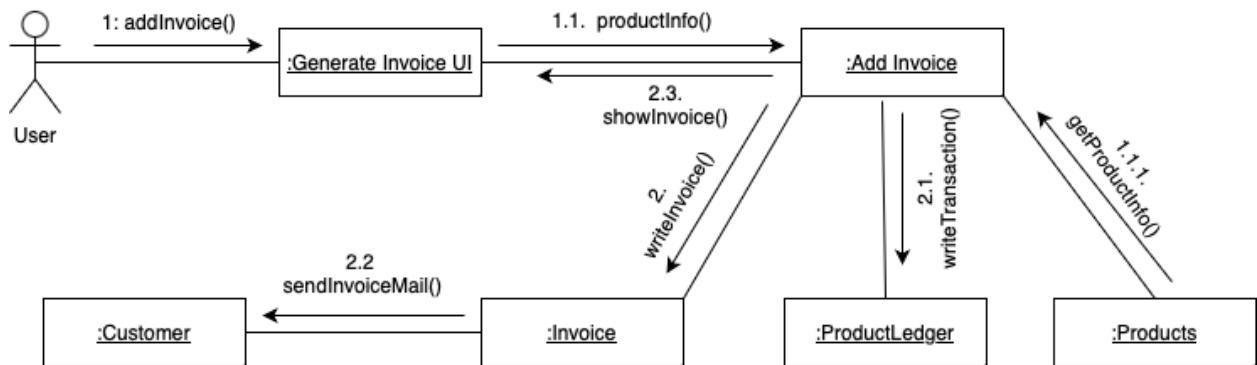


Figure 21: Communication diagram for generating invoices.

c) Sequence Diagram

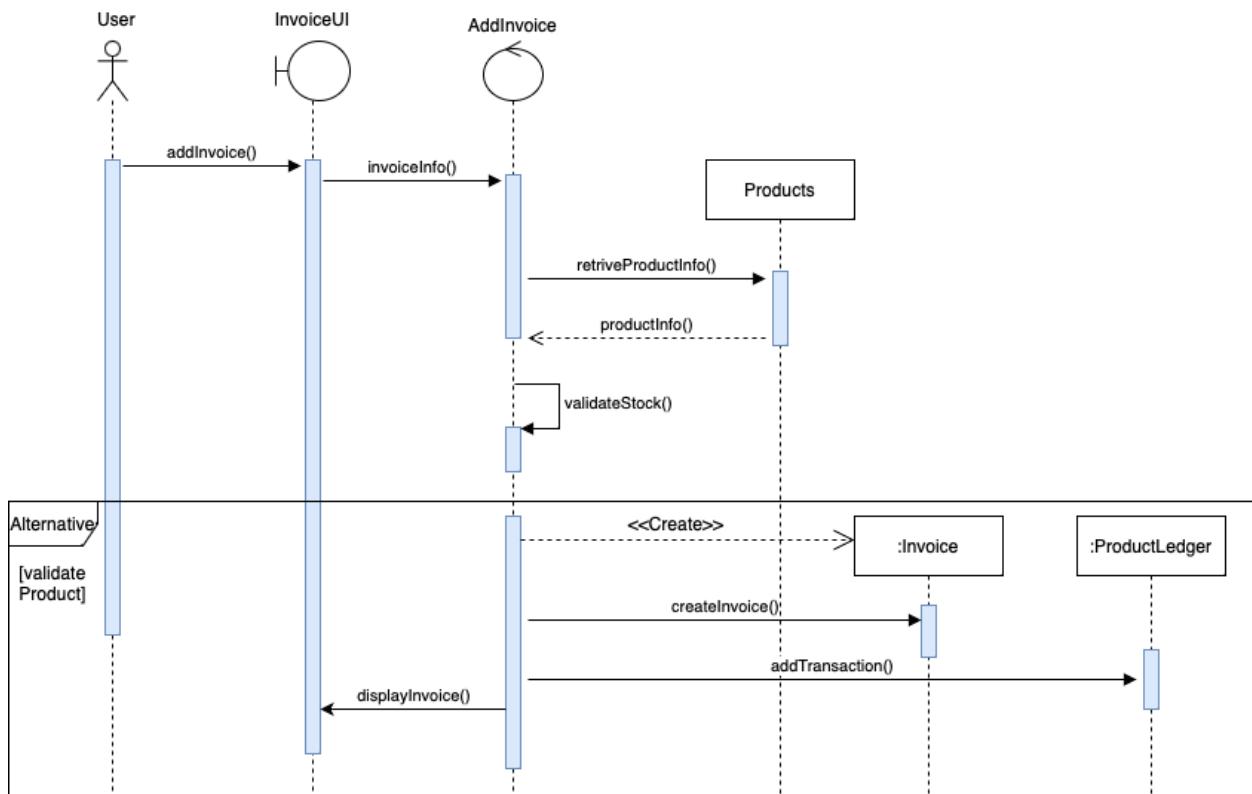


Figure 22: Sequence diagram for generating invoices

3.6.3.4. Add Packages to Invoices

a) Data Flow Diagram

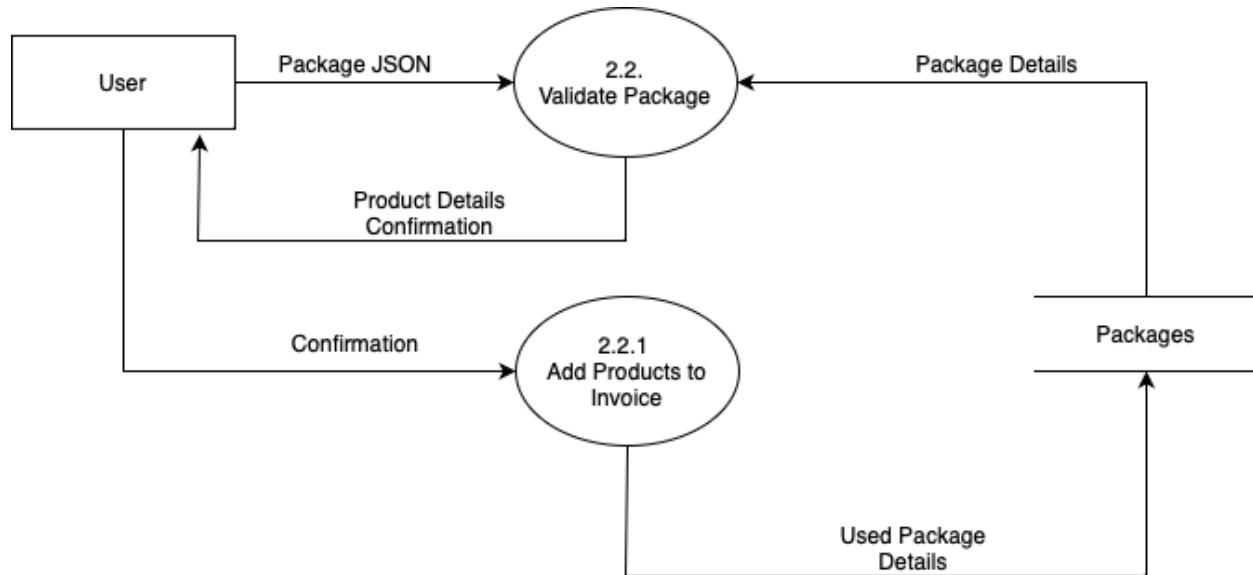


Figure 23: DFD Level 2 for adding packages to invoices.

b) Communication Diagram

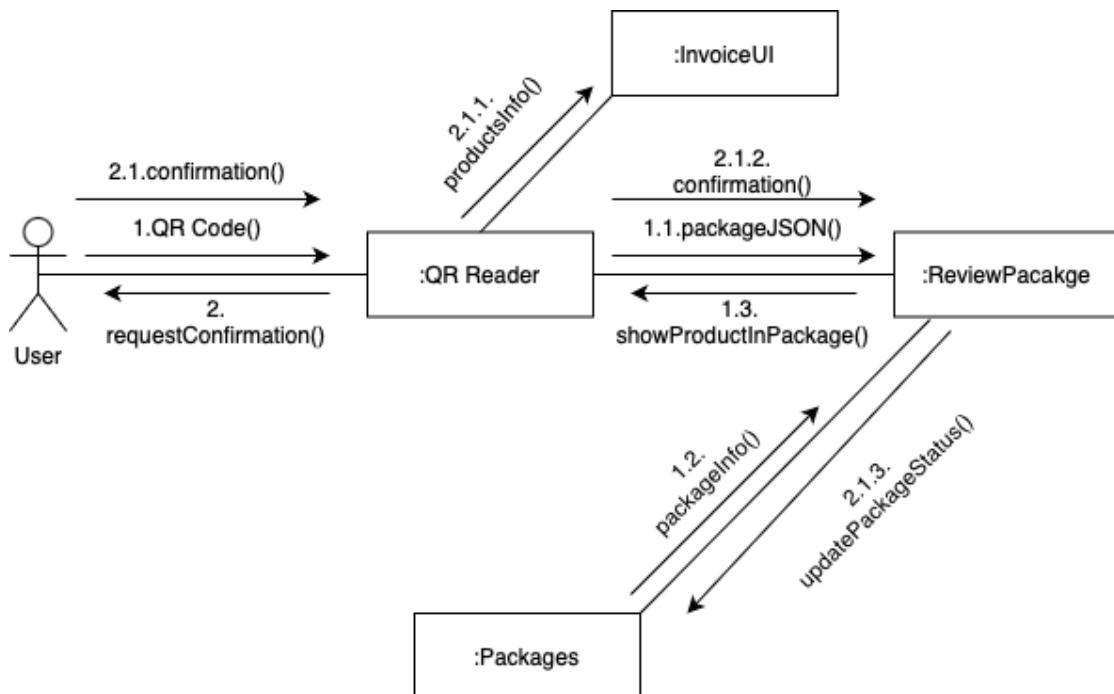


Figure 24: Communication diagram for adding packages to invoices.

3.6.3.5. Mark Attendance for employee

a) Data Flow Diagram

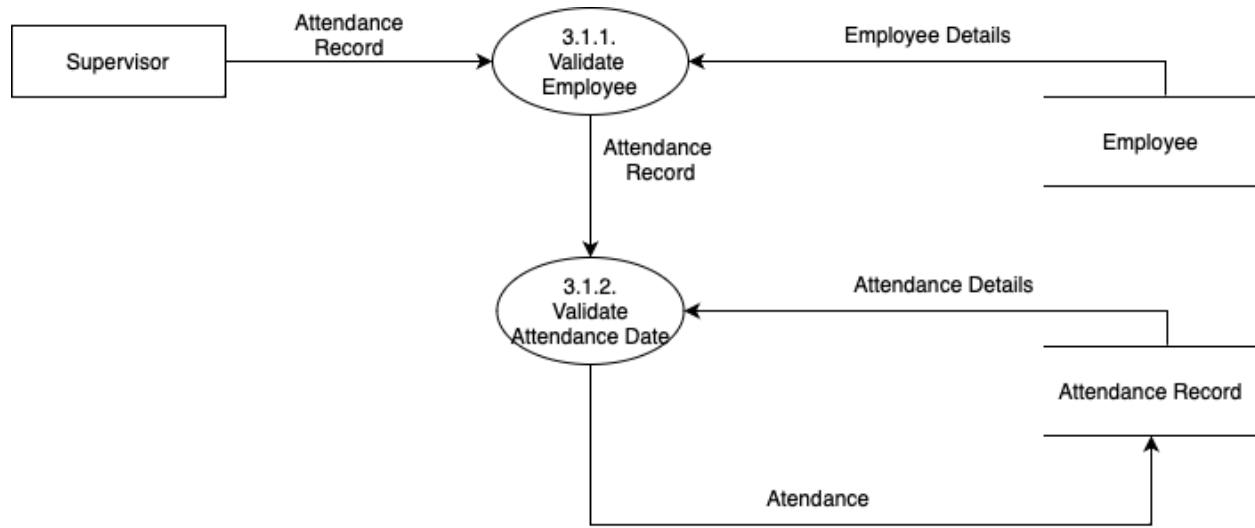


Figure 25: DFD Level 2 for adding attendance for employee.

b) Communication Diagram

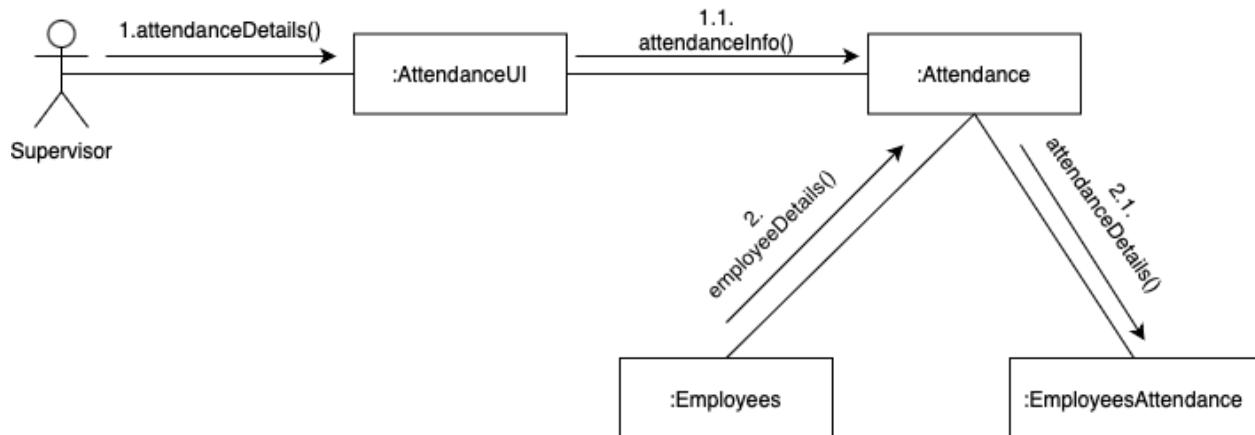


Figure 26: Communication diagram for adding attendance for employee

c) Sequence Diagram

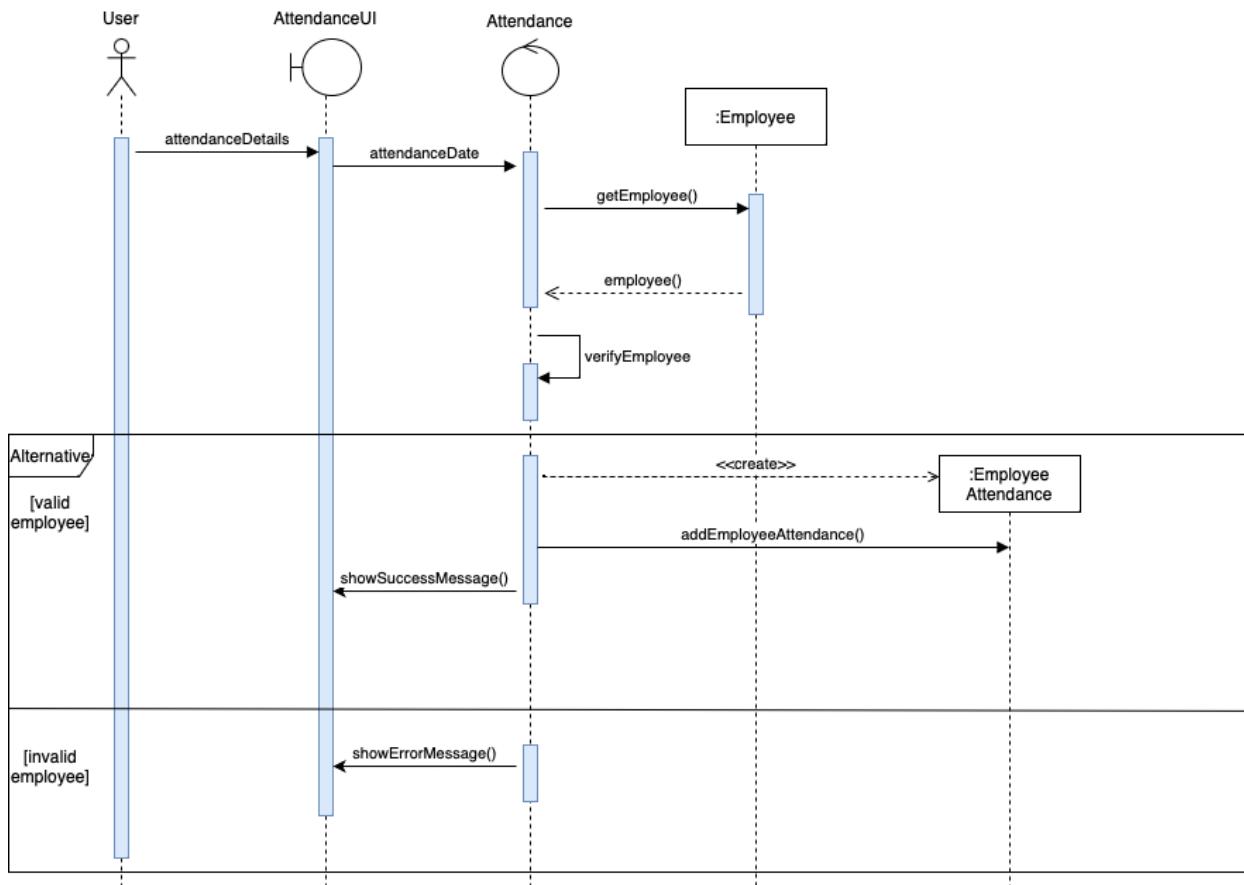


Figure 27: Sequences diagram for adding attendance for employee.

3.6.3.6. View Sales Report

a) Data Flow Diagram

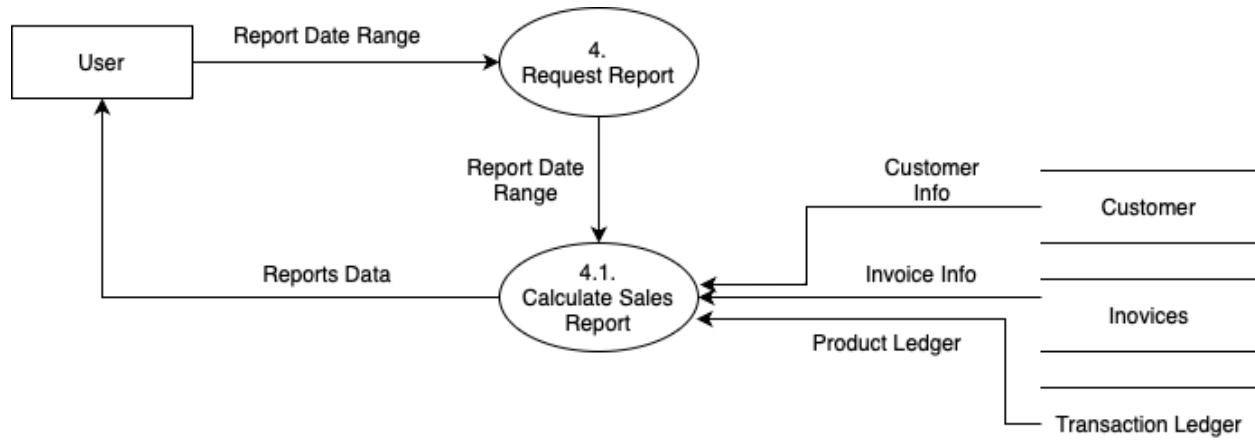


Figure 28: DFD Level 2 for generating sales report.

b) Communication Diagram

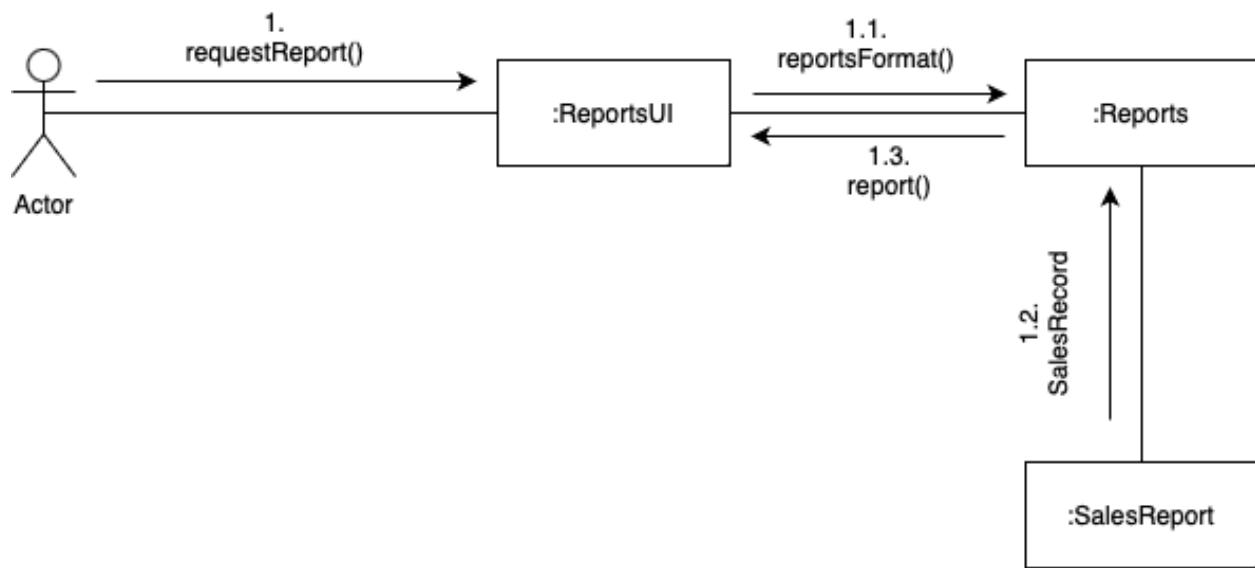


Figure 29: Communication diagram for generating sales report.

c) Sequence Diagram

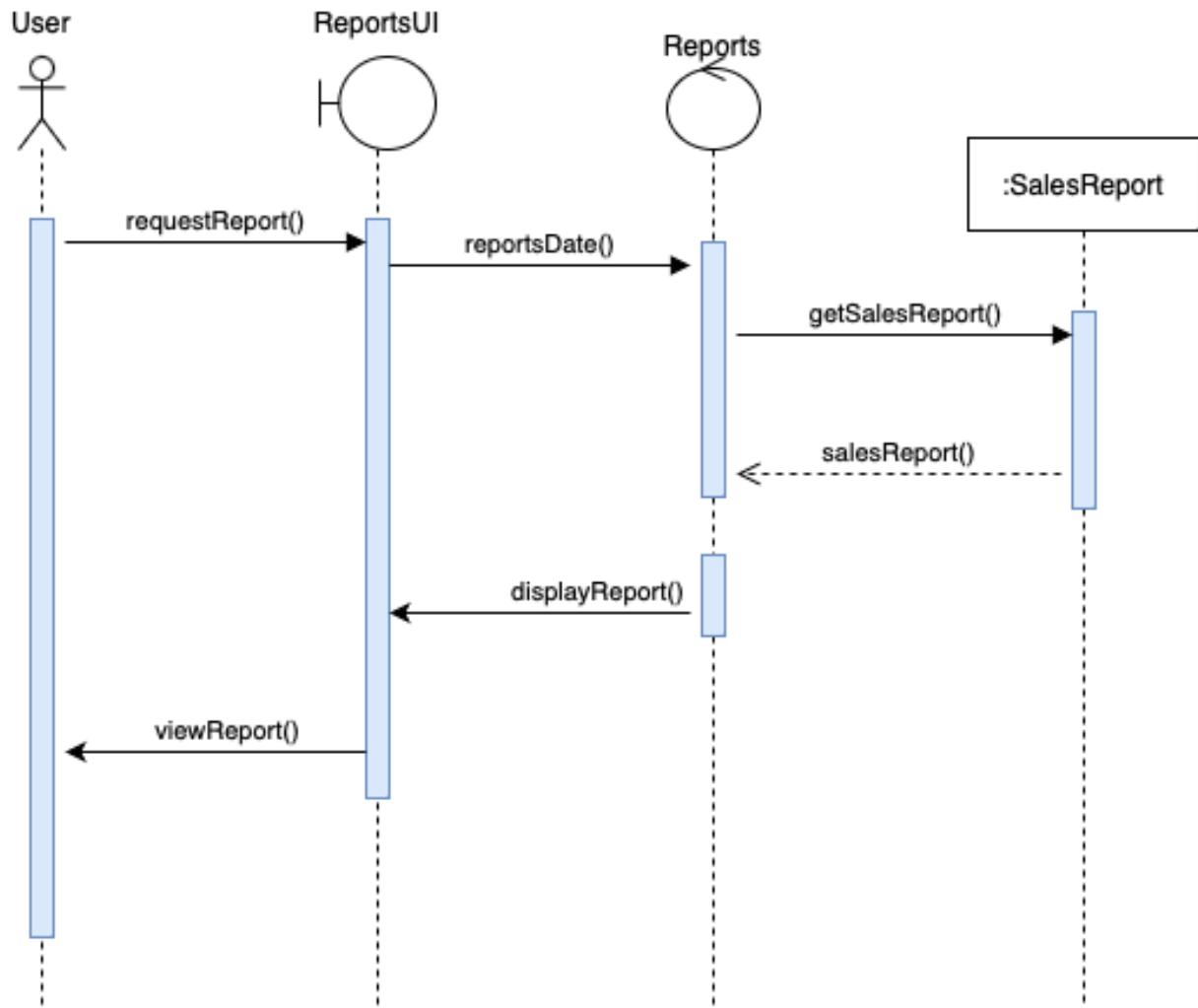


Figure 30: Sequence diagram for generating sales report.

3.6.3.7. Importing data from CSV file

a) Data Flow Diagram

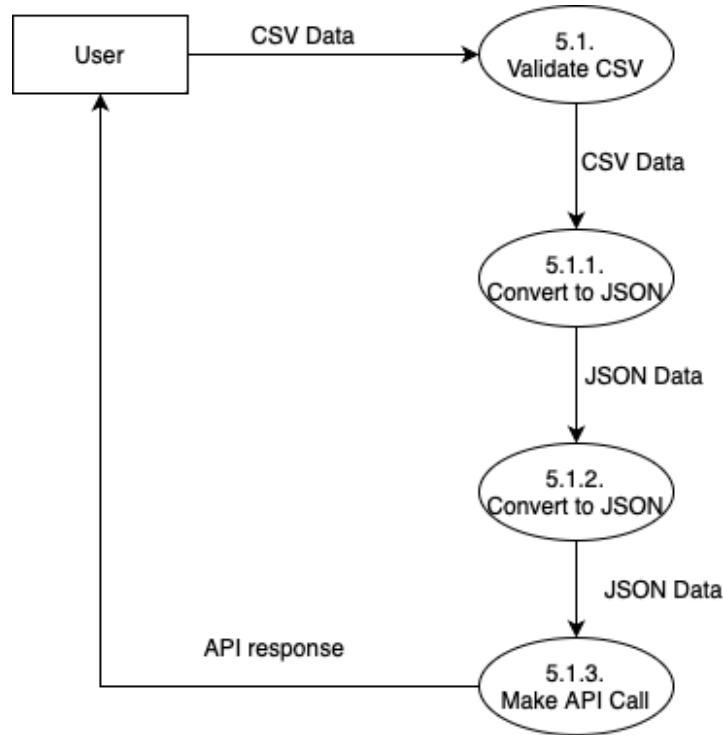


Figure 31: DFD Level 2 for importing data from csv.

b) Communication Diagram

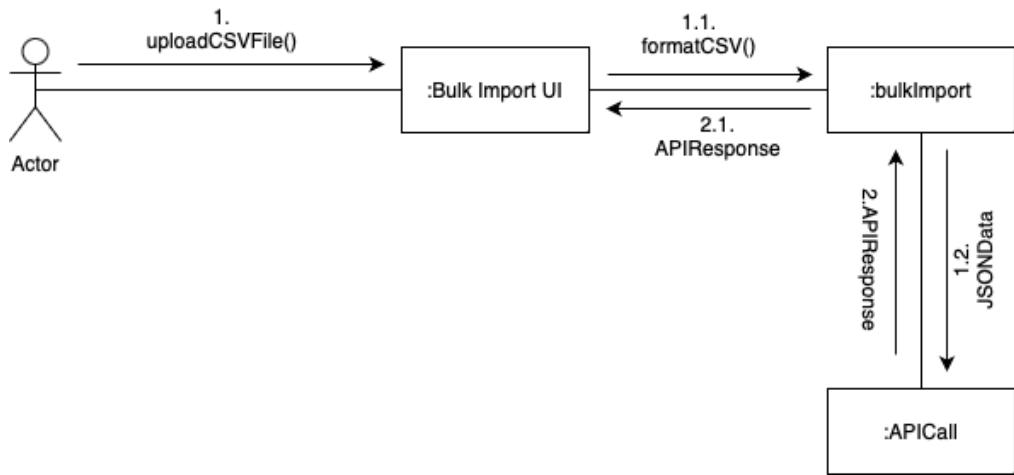


Figure 32: Communication diagram for importing data from csv.

c) Sequence Diagram

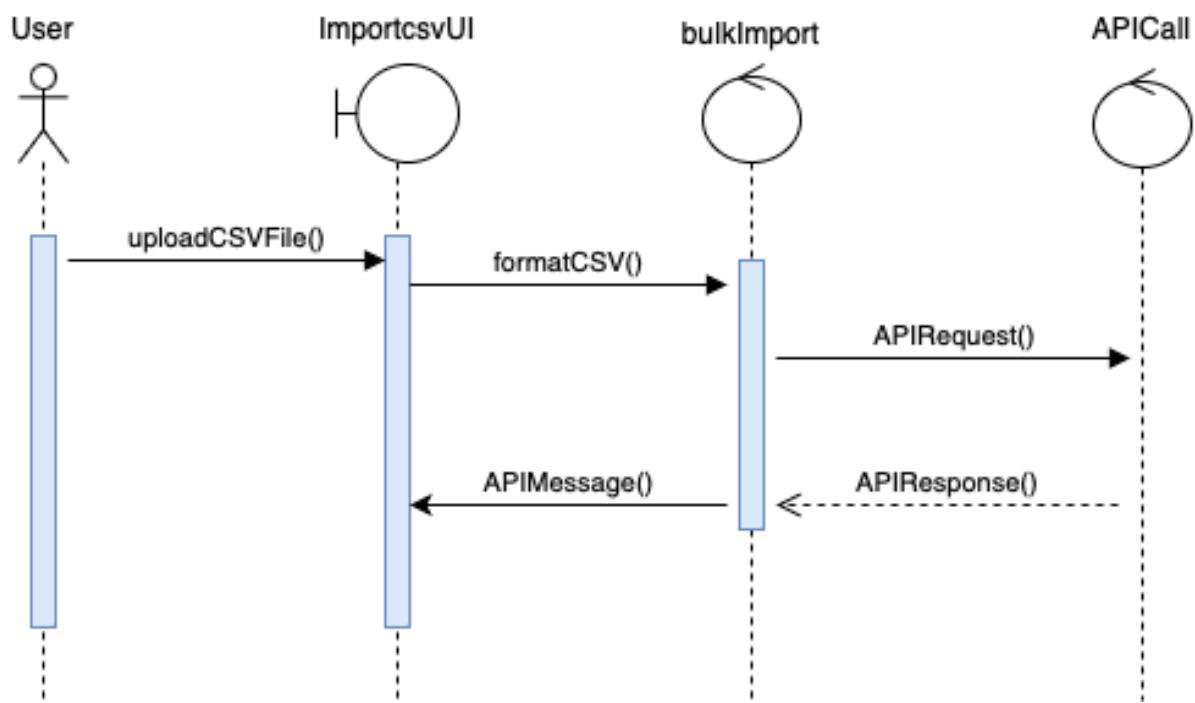


Figure 33: Sequence diagram for importing data for csv.

3.6.3.8. Adding Employee

a) Data Flow Diagram

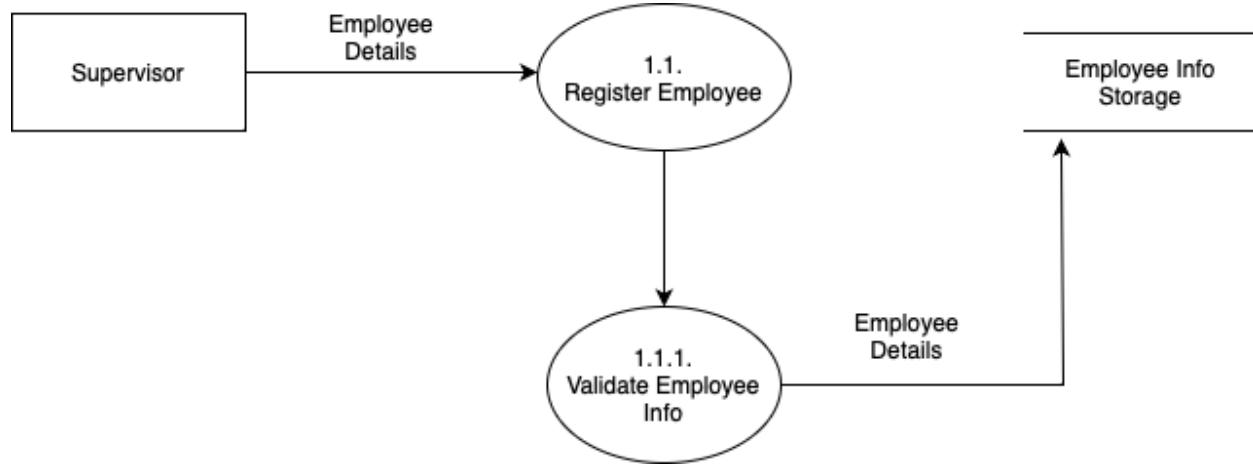


Figure 34: DFD Level 2 for adding new employees.

b) Communication Diagram

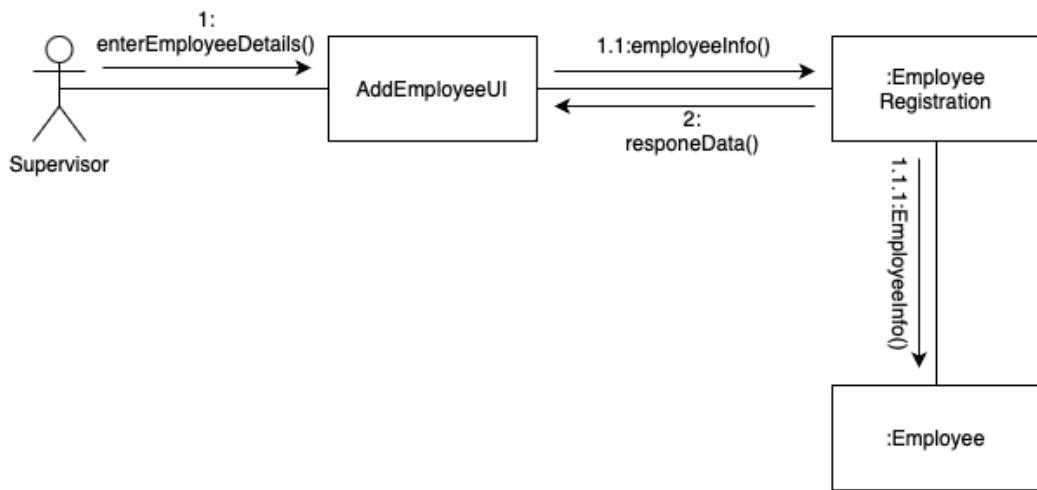


Figure 35: Collaboration Diagram for adding employee.

c) Sequence Diagram

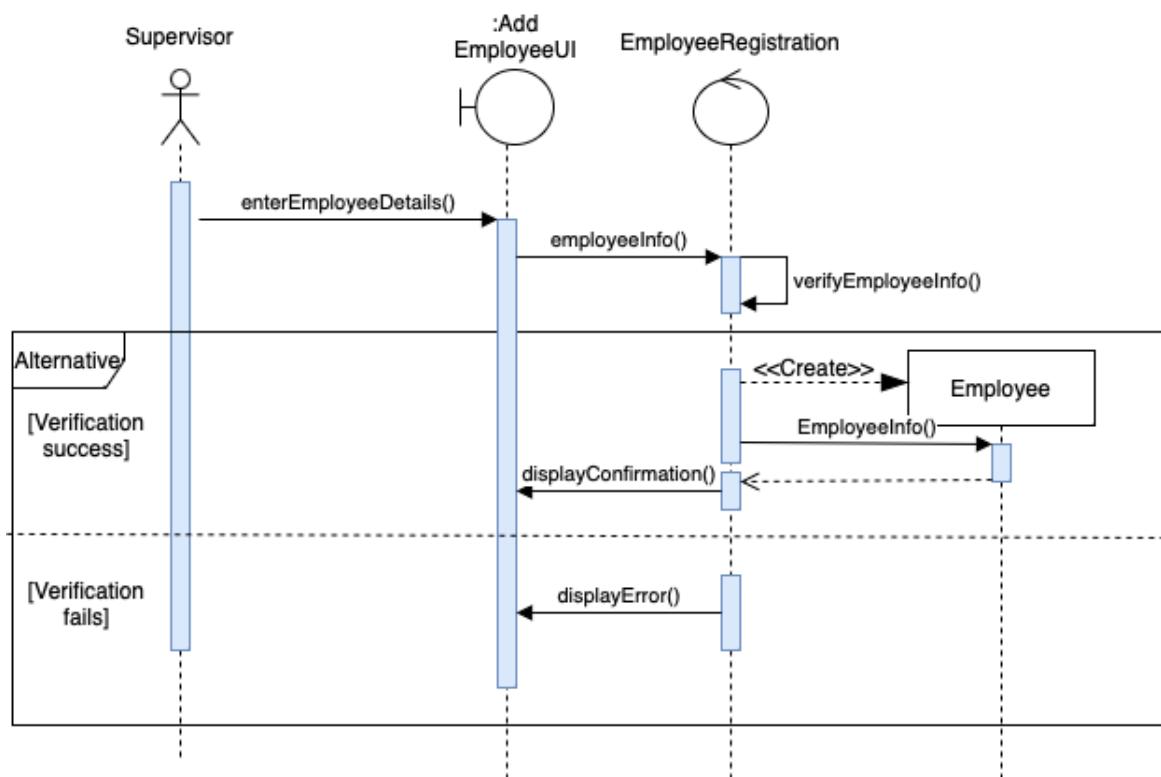


Figure 36: Sequence diagram for adding employee.

3.6.3.9. Generating QR Code for Packages

a) Data Flow Diagram

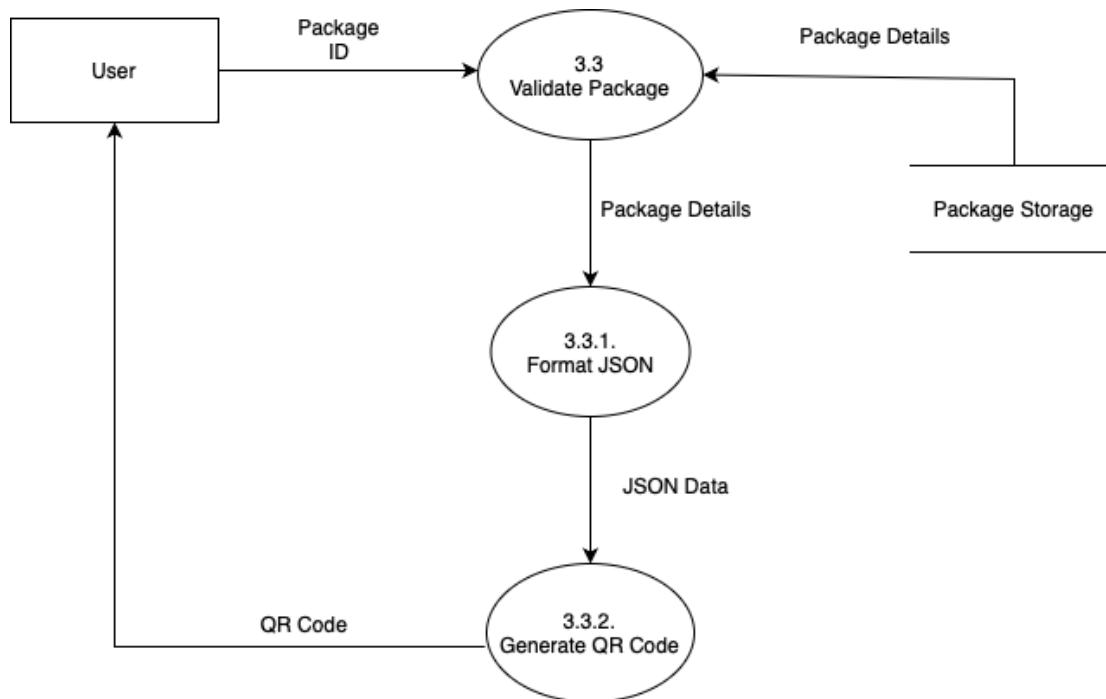


Figure 37: DFD Level 2 for generating QR Code.

b) Communication Diagram

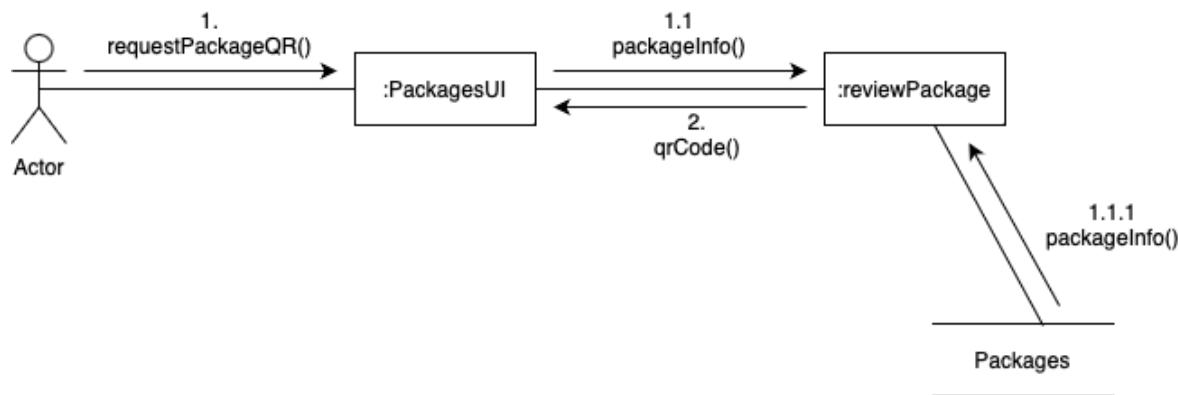


Figure 38: Communication Diagram for generating QR Code

c) Sequence Diagram

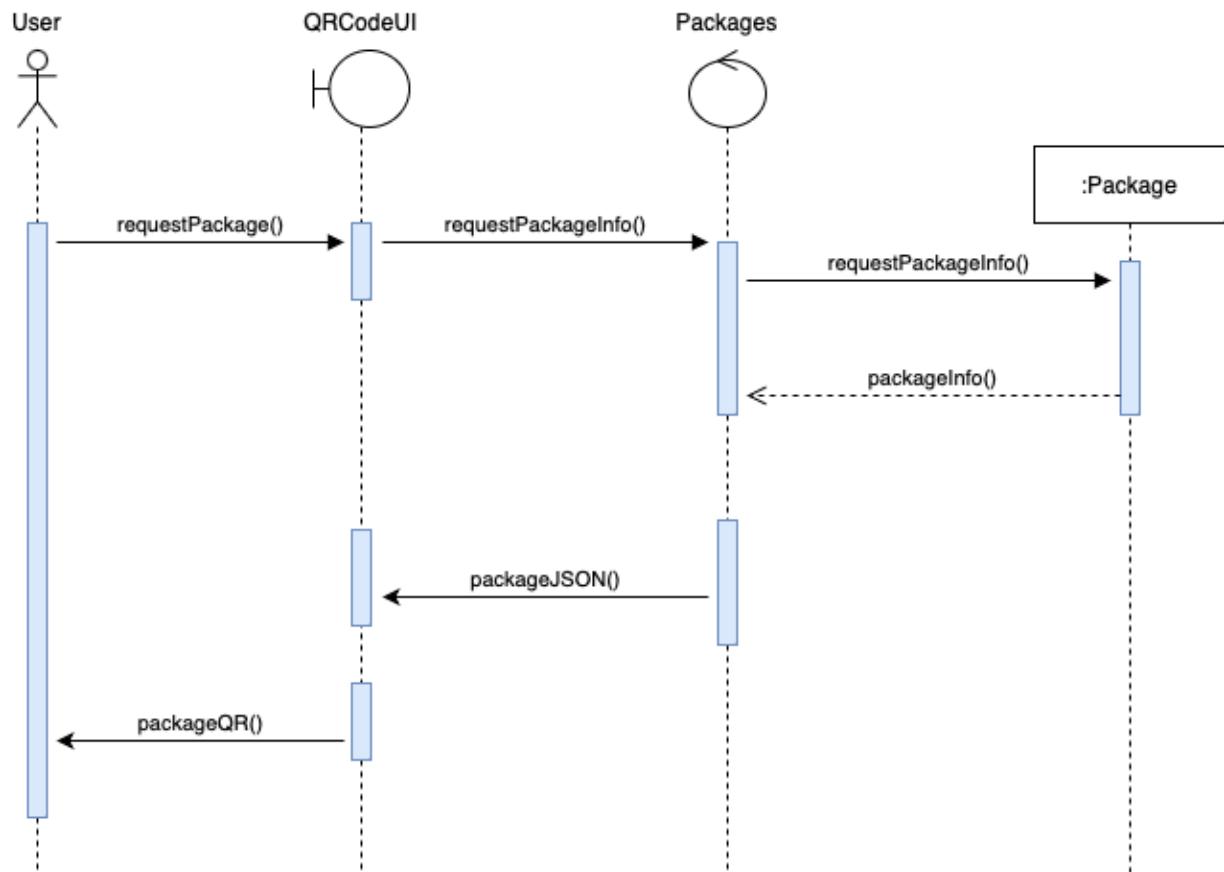


Figure 39: Sequence diagram for generating QR code.

3.6.4. Wireframes Developed

Add Customer

Company Logo

Search

Dashboard

Customer

Products

Invoices

Transaction

Packages

Add Employee

Employee Name

Phone

PAN

Location

Salary

Joined Date

/ /

Add

Cancel

This wireframe illustrates the user interface for adding a new employee. The left sidebar provides navigation between various application modules. The main content area is titled 'Add Employee' and contains several input fields: Employee Name, Phone, PAN, Location, Salary, and Joined Date. The Joined Date field includes a small calendar icon. At the bottom right, there are 'Add' and 'Cancel' buttons.

Figure 40: Add Employee Wireframe

Add Invoice

 Company Logo	<input type="text" value="Search"/> [Search icon]	[Bell icon] [User icon] [Exit icon]												
Dashboard	<h2>Add Invoice</h2>													
Customer	<input type="text"/> [Add icon]	Date: <input type="text"/> [Calendar icon]												
Products	<input type="text"/>													
Invoices	Items													
Packages	<table border="1"><thead><tr><th>#</th><th>Name</th><th>Item Code</th><th>Quantity</th><th>Price</th><th>Amount</th></tr></thead><tbody><tr><td colspan="6"><input type="text"/></td></tr></tbody></table>	#	Name	Item Code	Quantity	Price	Amount	<input type="text"/>						Total <input type="text"/> Discount <input type="text"/> VAT <input type="text"/> Grand Total <input type="text"/> [Add] [Cancel]
#	Name	Item Code	Quantity	Price	Amount									
<input type="text"/>														

Figure 41: Add Invoice Wireframe

Add Package

 Company Logo	<input type="text" value="Search"/>	  		
Dashboard	<h3>Add Package</h3>			
Customer				
Products				
Invoices				
Packages				
	<p>Items</p> <table border="1"><tr><td>#</td><td>Name</td></tr></table>		#	Name
#	Name			
	<p>Date: / / </p>			
	<table border="1"><tr><td>Price</td><td>Amount</td></tr></table>	Price	Amount	Total <input type="text"/>
Price	Amount			
	<p>Print</p>			
	<p>Package Added Successfully</p>  <p>QR Code for the package</p>			
	<p>Add Cancel</p>			

Figure 42: Add Package Wireframe

Add Transaction

 Company Logo	<input type="text" value="Search"/>	  
Dashboard	<h3>Add Transaction</h3>	
Customer	<input data-bbox="535 451 806 487" type="text"/>	Product ID <input type="text" value="2079/07/26"/> 
Products	<input data-bbox="535 566 1372 601" type="text"/>	Amount
Invoices	<input data-bbox="535 658 1372 694" type="text"/>	Description
Transaction	<input data-bbox="535 694 1372 730" type="text"/>	
Packages	<input type="button" value="Add"/> <input type="button" value="Cancel"/>	

Figure 43: Add Transaction Wireframe

Add Customer

<input type="checkbox"/> Company Logo	<input type="text"/> Search	<input type="button"/>
Add Customer		
Dashboard	Customer Name	
Customer	<input type="text"/>	
Products	Phone	PAN
Invoices	<input type="text"/>	<input type="text"/>
Packages	Location	
	<input type="text"/>	
	Email	Opening Balance
	<input type="text"/>	<input type="text"/>
	<input type="button"/> Add	<input type="button"/> Cancel

Figure 44: Add Customer Wireframe

Add Products

 Company Logo	<input type="text"/> Search	  
Dashboard	<h3>Add Products</h3>	
Customer	<input type="text"/> Product Name	 Upload Image
Products	<input type="text"/> Product ID	
Invoices	<input type="text"/> Category	
Packages	<input type="text"/> Opening Stock	
	<input type="text"/> Selling Price <input type="text"/> Cost Price	
	<input type="button" value="Add"/> <input type="button" value="Cancel"/>	

Figure 45: Add Products Page

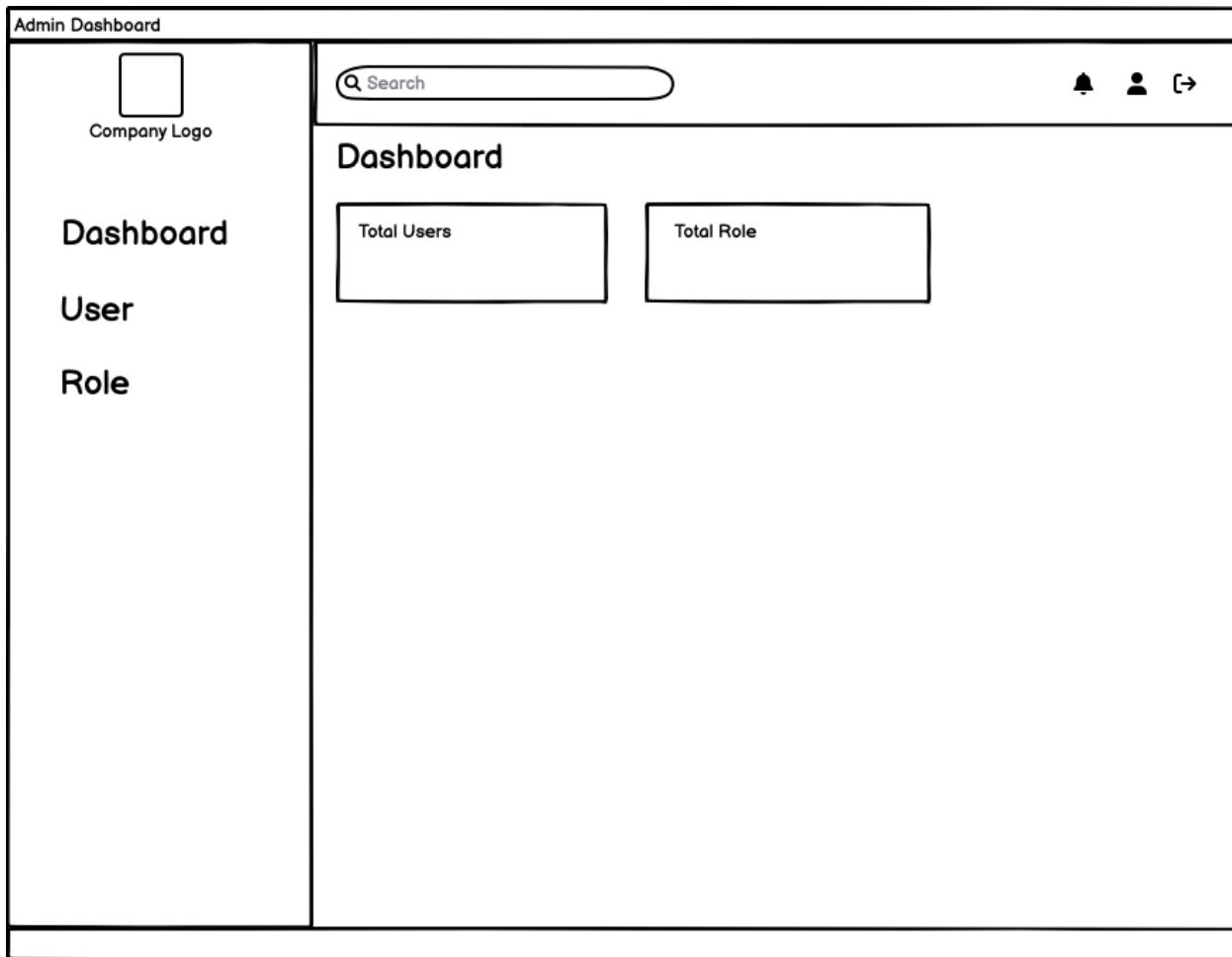


Figure 46: Admin Dashboard Wireframe

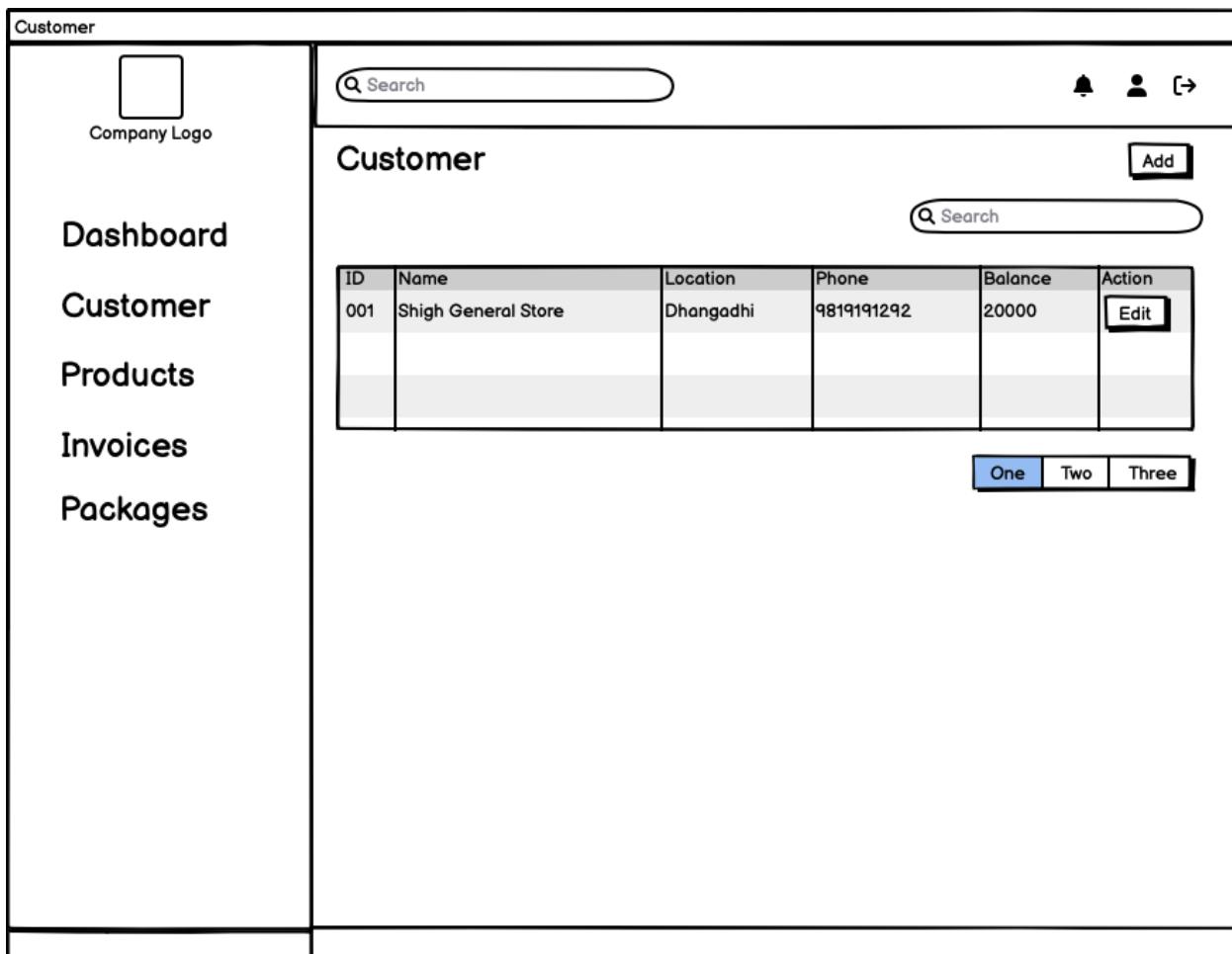


Figure 47: Customer Wireframe

Edit Customer

 Company Logo	<input type="text" value="Search"/>	  
<h2>Edit Customer</h2>		
Dashboard	Customer Name <input type="text" value="Lorem"/>	
Customer	Phone <input type="text" value="Lorem"/>	PAN <input type="text" value="Lorem"/>
Products	Location <input type="text" value="Lorem"/>	
Invoices	Email <input type="text" value="Lorem"/>	Opening Balance <input type="text" value="Lorem"/>
Packages		
		<input type="button" value="Update"/> <input type="button" value="Cancel"/>

Figure 48: Edit Customer Wireframe

Edit Invoice

 Company Logo	<input type="text" value="Search"/>	  												
Dashboard	<h2>Edit Invoice</h2>													
Customer	Customer Name <input type="text" value="Lorem"/> 	Date: <input type="text" value="2079 /07/26"/> 												
Products	Location <input type="text" value="Lorem"/>													
Invoices	Items													
Packages	<table border="1"><thead><tr><th>#</th><th>Name</th><th>Item Code</th><th>Quantity</th><th>Price</th><th>Amount</th></tr></thead><tbody><tr><td colspan="6"></td></tr></tbody></table>	#	Name	Item Code	Quantity	Price	Amount							Total <input type="text" value="0"/> Discount <input type="text" value="0"/> VAT <input type="text" value="0"/> Grand Total <input type="text" value="0"/> <input type="button" value="Add"/> <input type="button" value="Cancel"/>
#	Name	Item Code	Quantity	Price	Amount									

Figure 49: Edit Invoice Wireframe

Edit Product

 Company Logo	<input type="text" value="Search"/>	  
Dashboard	<h3>Edit Products</h3>	
Customer	Product Name	
Products	<input type="text" value="Lorem"/>	Upload Image
Invoices	Product ID	
Packages	<input type="text" value="Lorem"/>	
	Category	
	<input type="text" value="Lorem"/>	
	Opening Stock	
	<input type="text" value="Lorem"/>	
	Selling Price	Cost Price
	<input type="text" value="Lorem"/>	<input type="text" value="Lorem"/>
		Update Cancel

Figure 50: Edit Product Wireframe

Edit Role

 Company Logo	<input type="text" value="Search"/>	  
Dashboard	<h2>Edit Role</h2>	
User	Role Name	
Role	<input type="text" value="ram_role"/>	
	Available Roles	
	<input checked="" type="checkbox"/> Products	<input checked="" type="checkbox"/> Employees
	<input type="checkbox"/> Money Transaction	<input type="checkbox"/> Customers
	<input type="button" value="Update"/> <input type="button" value="Cancel"/>	

Figure 51: Edit Role WIreframe

Edit User

Company Logo

Dashboard

User

Role

Search

Edit User

User Name

Password

Confirm Password

User Type

Role

Figure 52:Edit User Wireframe

Employee's Attendance

Company Logo	<input type="text"/> Search	Notification Bell User Profile [→]												
Dashboard	<h3>Attendance</h3> <p>Date <input type="text"/></p> <table border="1"><thead><tr><th>#</th><th>Name</th><th>Action</th></tr></thead><tbody><tr><td>1</td><td>Ram Babu</td><td>Present</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table> <p><input type="radio"/> All Present <input type="radio"/> All Absent</p> <p>Add Cancel</p>		#	Name	Action	1	Ram Babu	Present						
#	Name	Action												
1	Ram Babu	Present												
Products														
Transaction														
Employee														
Reports														

Figure 53: Add Attendance Wireframe

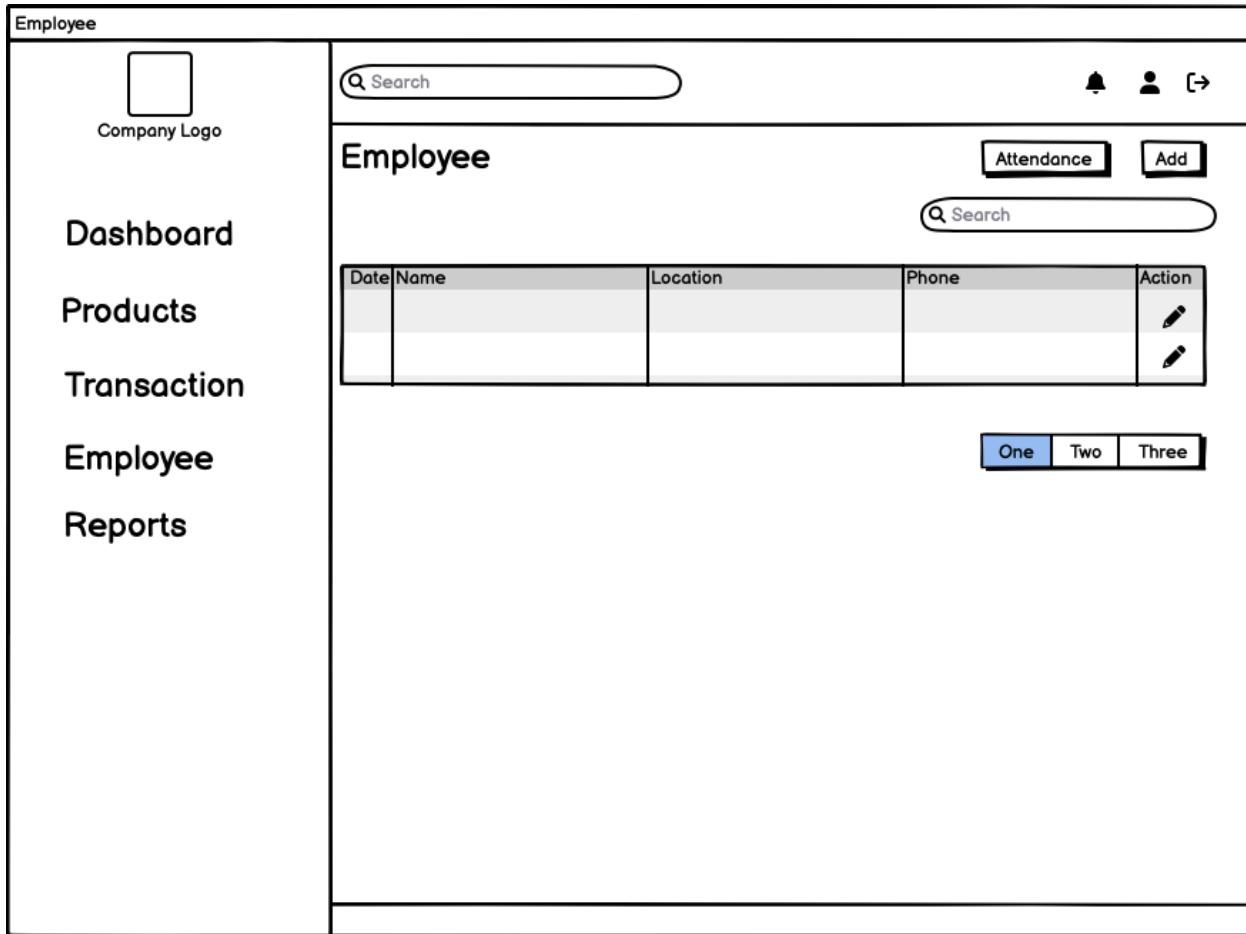


Figure 54: Employee Wireframe

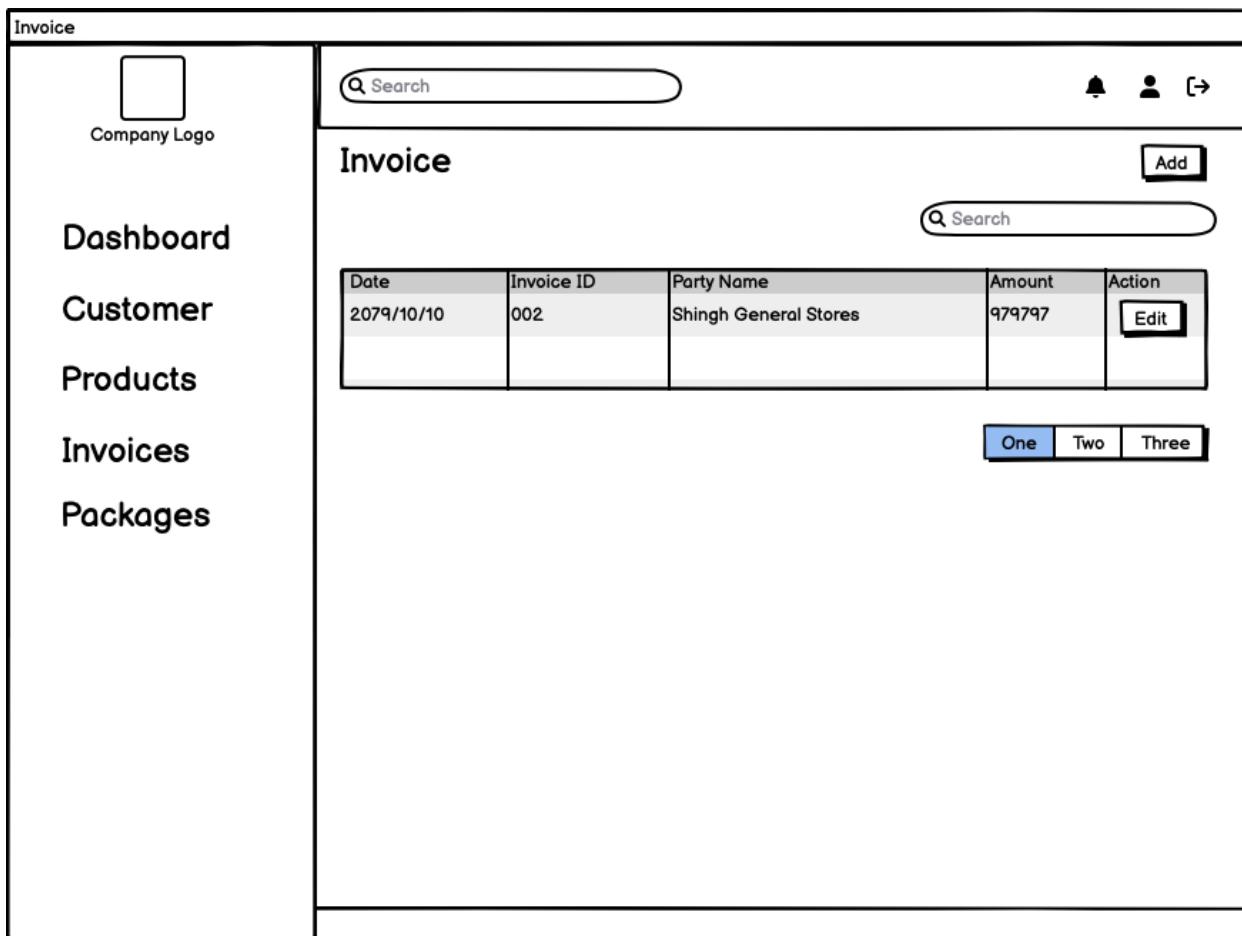


Figure 55: Invoice Wireframe

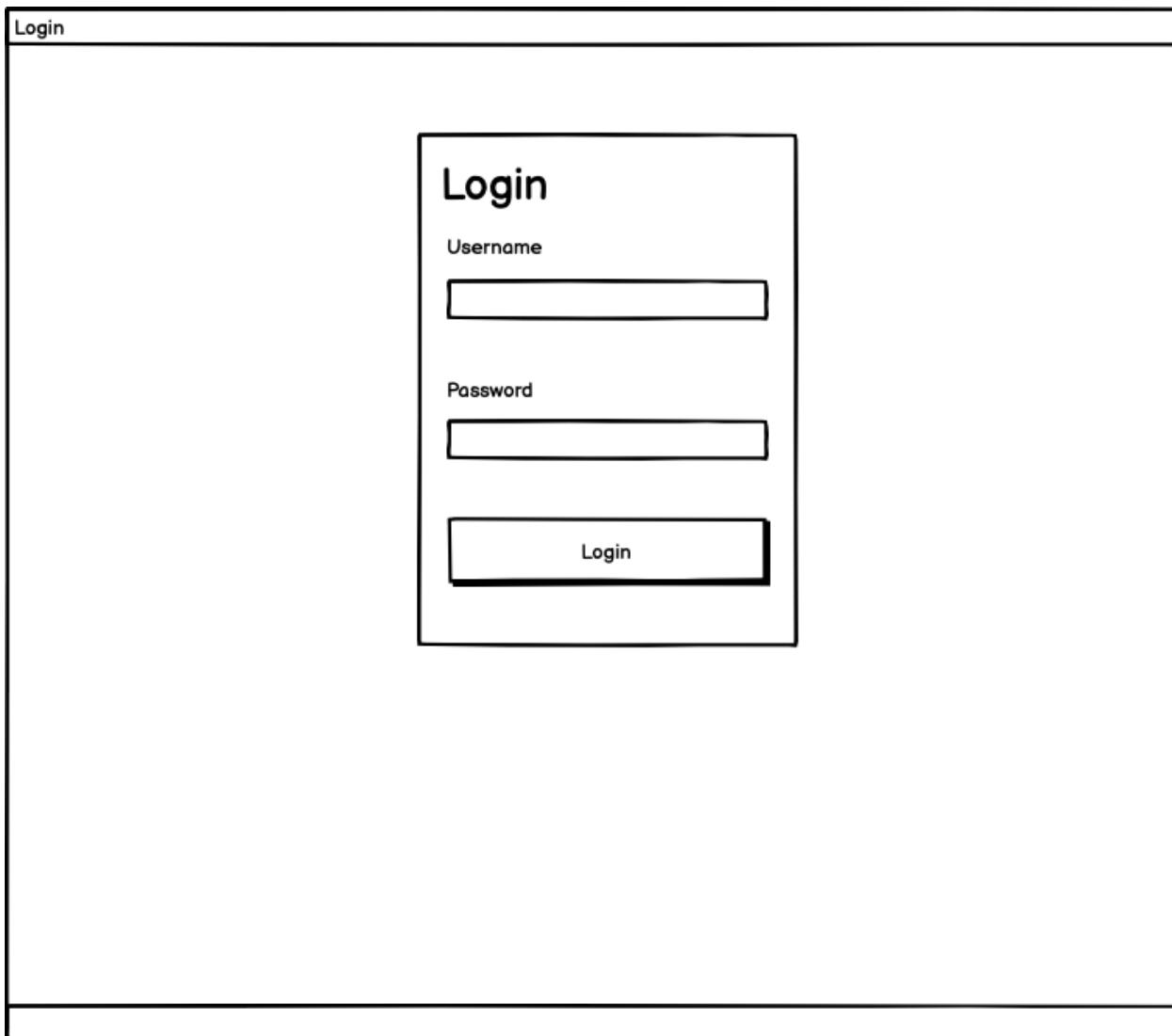


Figure 56: Login Wireframe

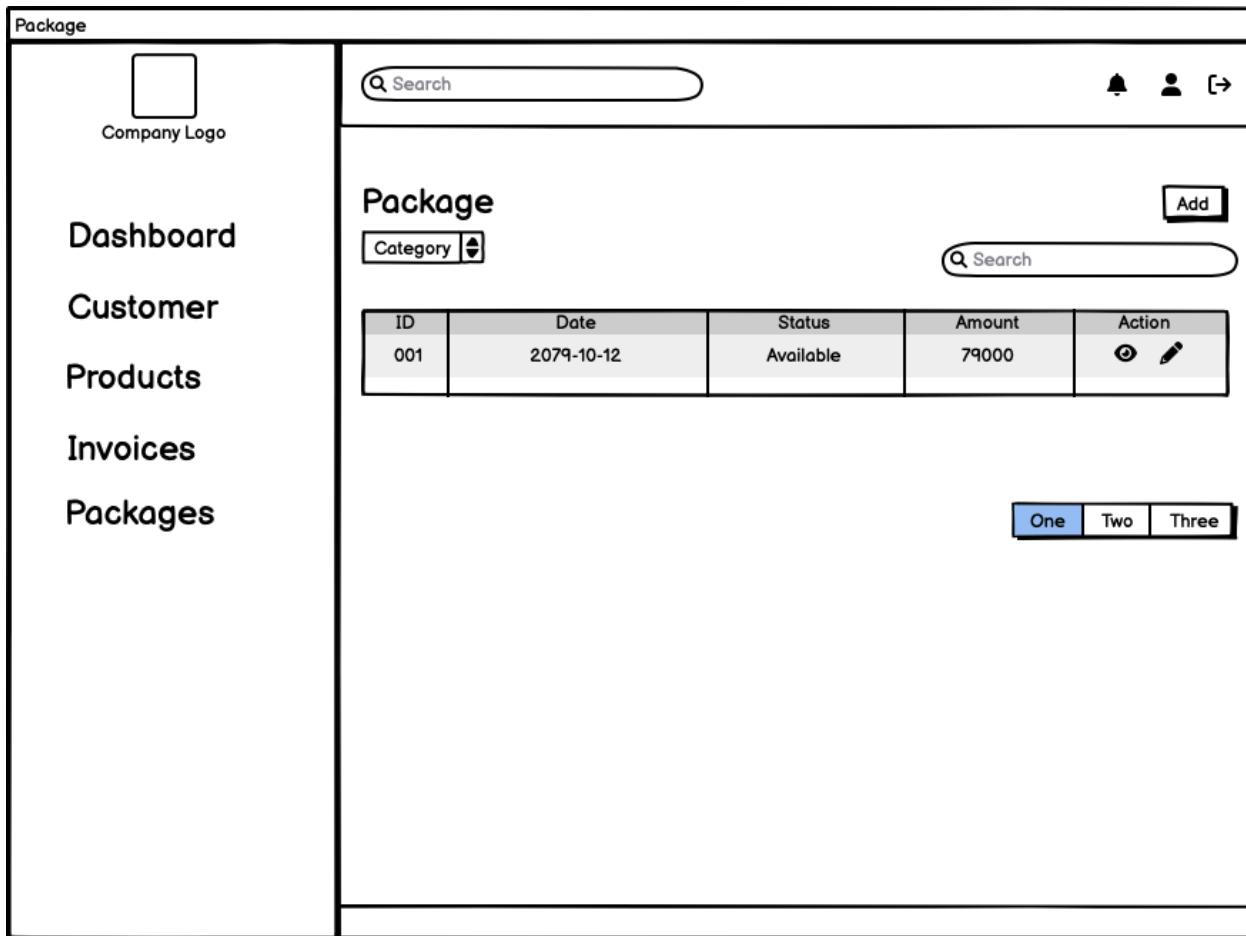


Figure 57: Production Wireframe

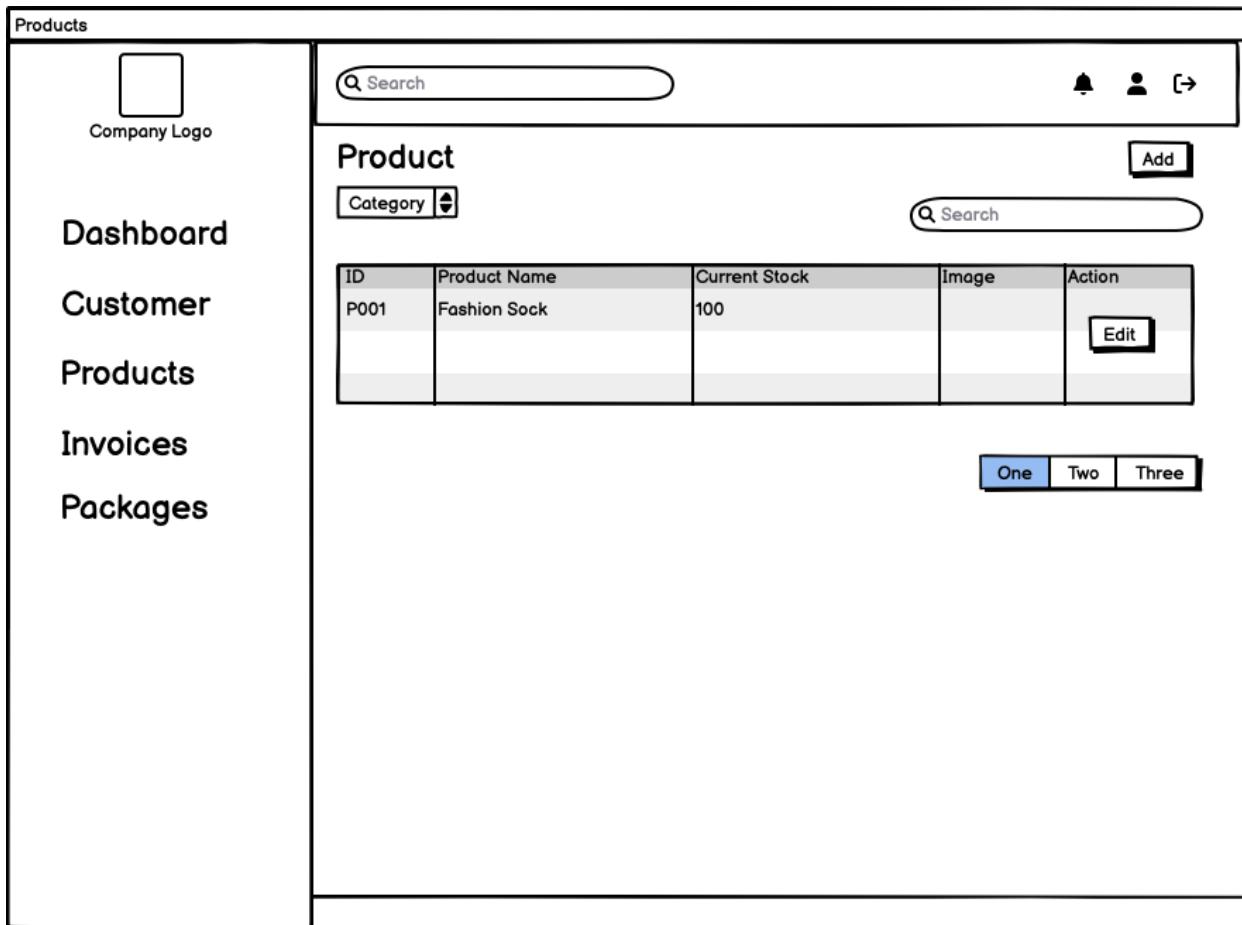


Figure 58: Product Wireframe

Report Generation

 Company Logo Dashboard Products Transaction Employee Reports	<input placeholder="Search" style="width: 400px; margin-bottom: 10px;" type="text"/> 🔔 🚙 ⏮ Generate Report <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <input type="text" value="Report Type"/> </div> <div style="flex: 1;"> <input type="text" value="Start Date"/> </div> <div style="flex: 1;"> <input type="text" value="End Date"/> </div> <div style="flex: 1;"> <input type="button" value="Generate"/> </div> </div> <table border="1" style="margin-top: 10px; width: 100%;"> <thead> <tr style="background-color: #f2f2f2;"> <th>S.No</th> <th>Product Name</th> <th>Quantity</th> <th>Price</th> <th>Amount</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Fashion Socks</td> <td>100</td> <td>300</td> <td>30000</td> </tr> <tr><td colspan="5" style="height: 100px;"></td></tr> </tbody> </table> <div style="text-align: right; margin-top: 10px;"> <input type="button" value="Back"/> <input type="button" value="Export"/> </div>	S.No	Product Name	Quantity	Price	Amount	1	Fashion Socks	100	300	30000																																			
S.No	Product Name	Quantity	Price	Amount																																										
1	Fashion Socks	100	300	30000																																										

Figure 59: Generate Report Wireframe

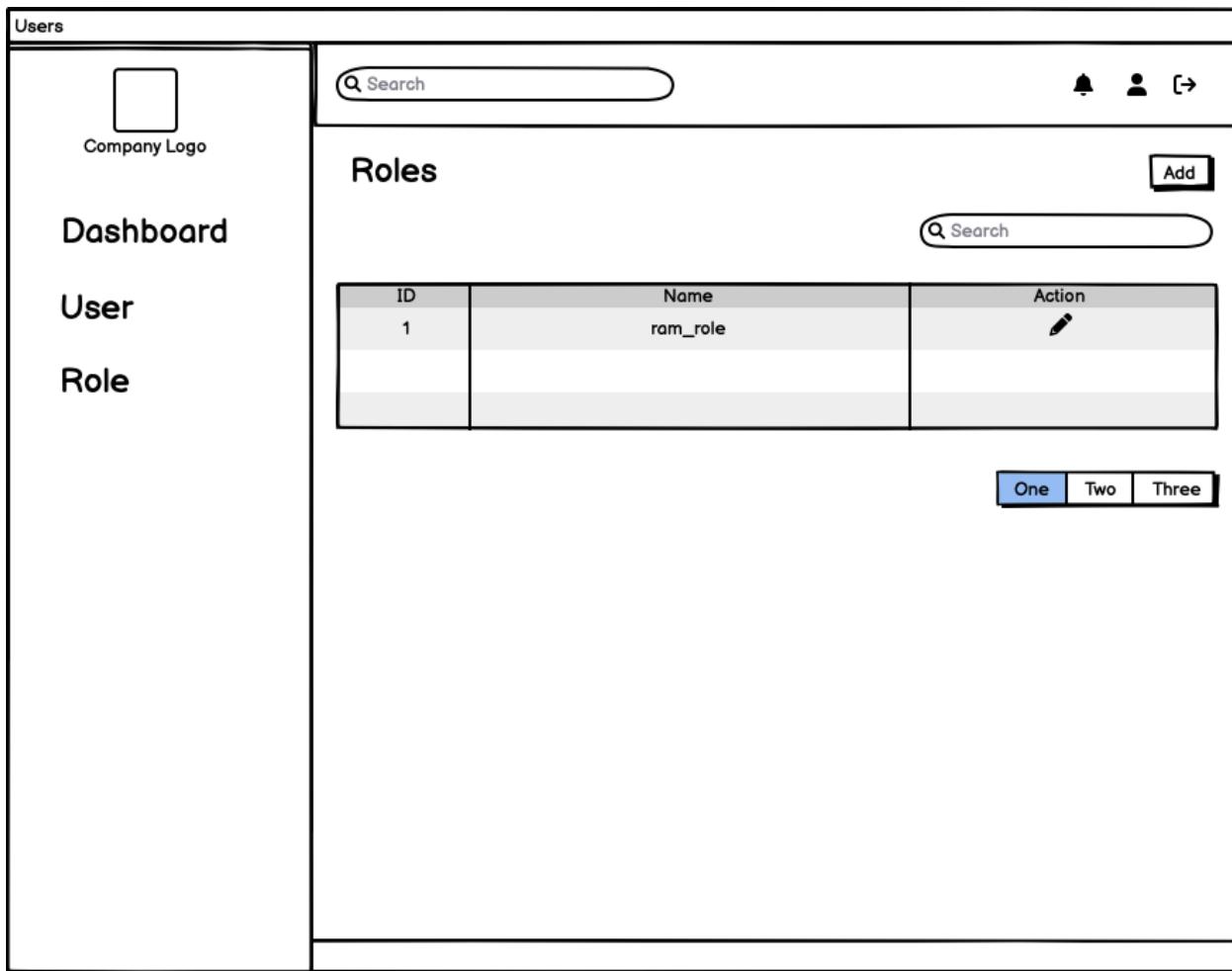


Figure 60: Role Wireframe

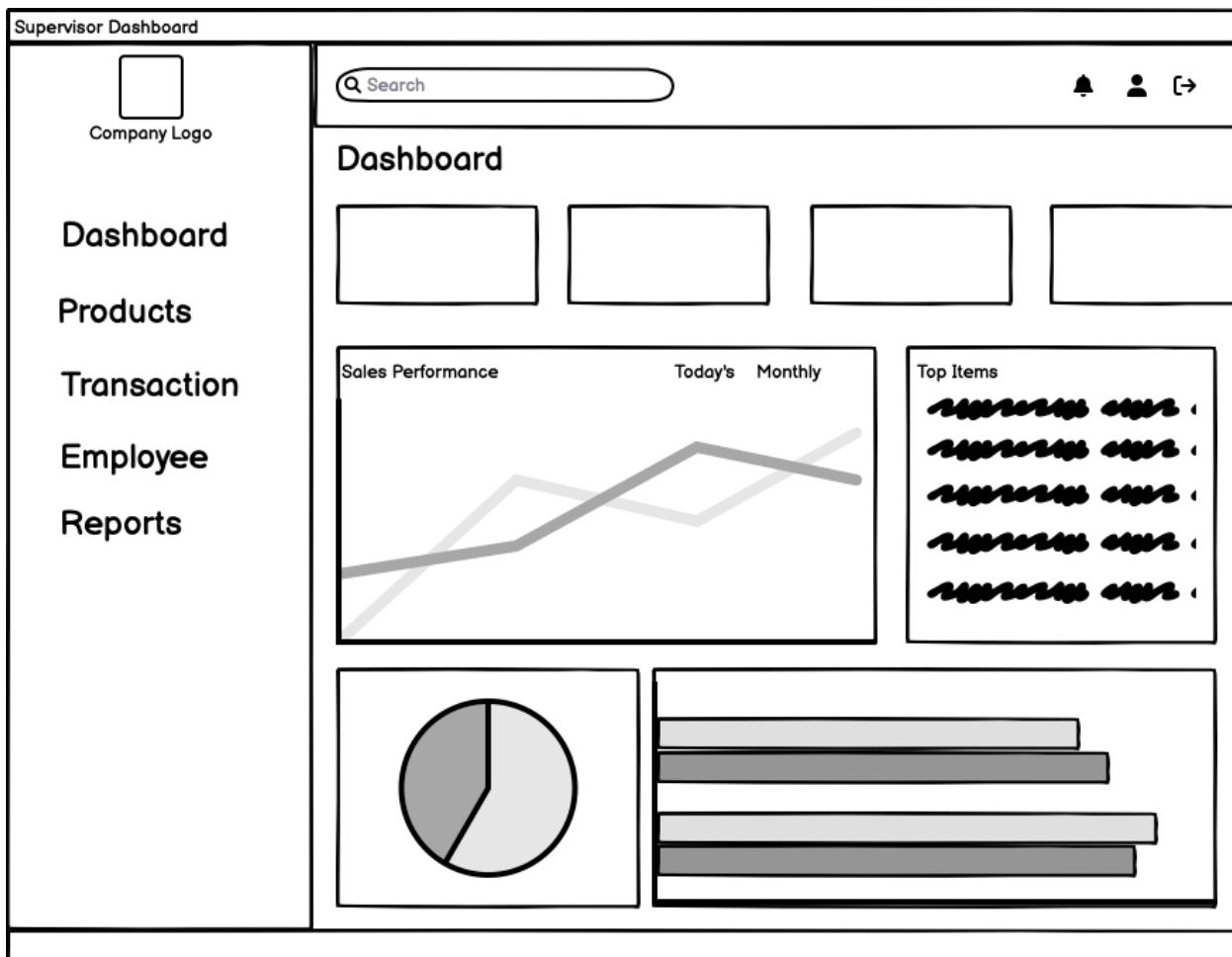


Figure 61: Supervisor Dashboard Wireframe

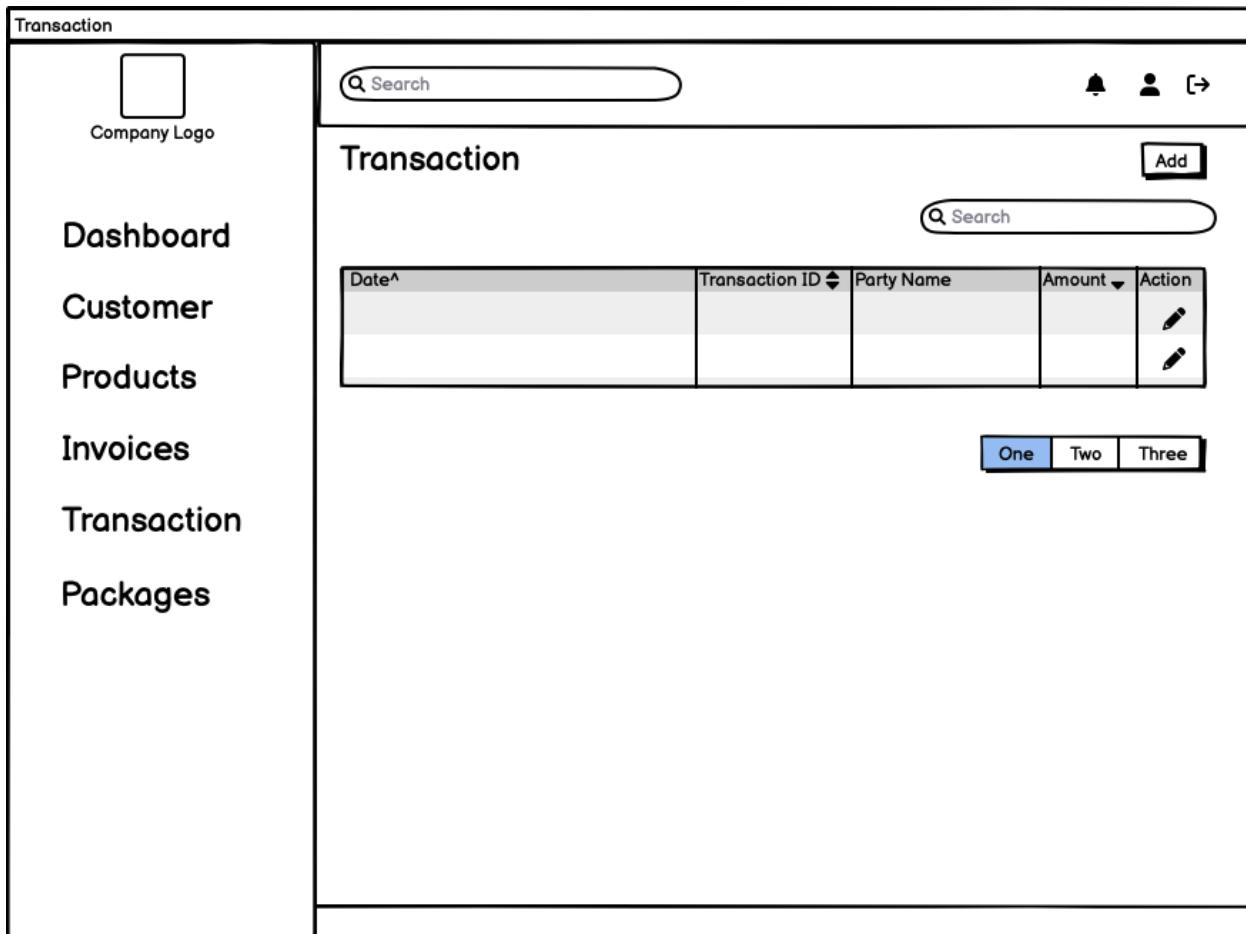
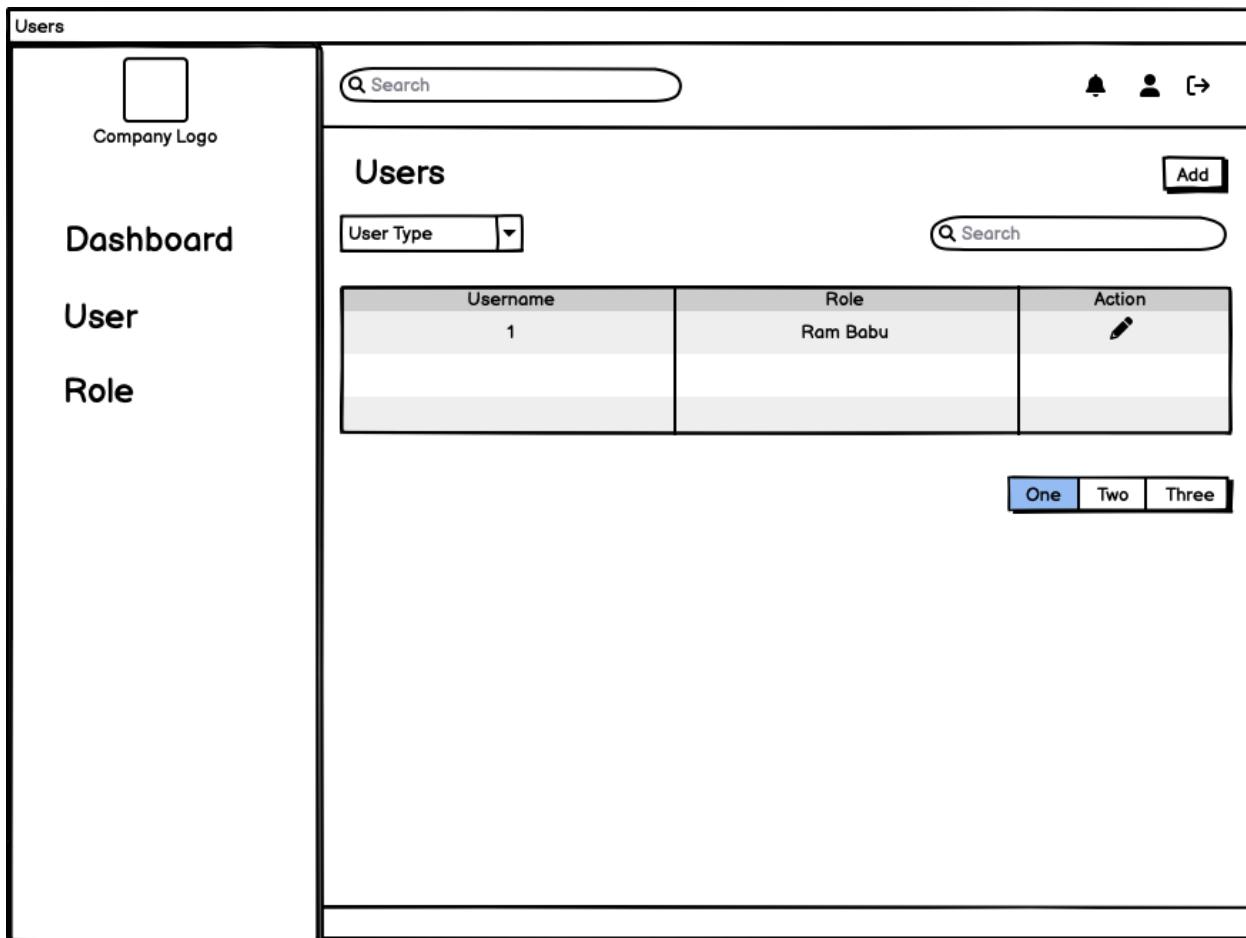


Figure 62: Transaction Wireframe



The wireframe for the 'Users' page is structured as follows:

- Header:** A top navigation bar with a 'Company Logo' placeholder, a search bar, and user profile icons.
- Left Sidebar:** A vertical sidebar containing three menu items: 'Dashboard', 'User', and 'Role'.
- Main Content Area:** A large area titled 'Users' featuring:
 - A dropdown menu for 'User Type'.
 - A search bar.
 - A table with columns: 'Username', 'Role', and 'Action'. The first row shows data: '1' in the Username column, 'Ram Babu' in the Role column, and a pencil icon in the Action column.
 - Three buttons at the bottom labeled 'One', 'Two', and 'Three'.

Figure 63: Users Wireframe

View Package

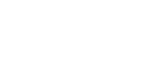
 Company Logo	<input type="text" value="Search"/>	  										
Dashboard	 Package Number 001	Date: 12 / 10 /2079 										
Customer	Items											
Products	<table border="1"><thead><tr><th>S.No</th><th>Product Name</th><th>Quantity</th><th>Price</th><th>Amount</th></tr></thead><tbody><tr><td>1</td><td>Fashion Socks</td><td>100</td><td>300</td><td>30000</td></tr></tbody></table>	S.No	Product Name	Quantity	Price	Amount	1	Fashion Socks	100	300	30000	Total: 30000
S.No	Product Name	Quantity	Price	Amount								
1	Fashion Socks	100	300	30000								
Invoices	 											
Packages	 	 										

Figure 64: Package Info Wireframe

3.7. Implementation

As the project was following proper SCRUM methodology, all the development performed in the project were break down to sprint and were completed following proper timeline developed in Gantt chart. Also, the iteration of developed Gantt Chart is included in appendix section.

3.7.1. Sprint 1 – User Management

3.7.1.1. Sprint Planning

As part of this project, user management was the very first task selected from product backlog. As per the product backlog, ability of the allowing user to securely access the application was by implementing login credentials. Also, the application would also allow to register users in the system.

a) Figma Mockup

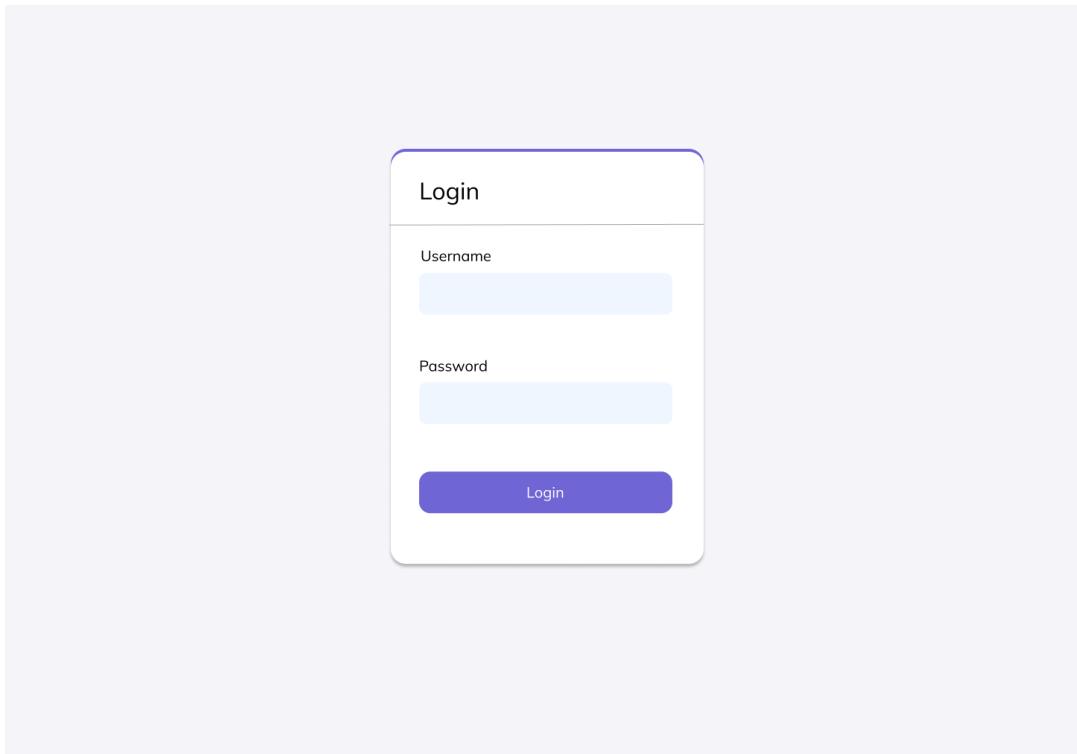


Figure 65: Figma Mockup for Login Page

The image shows a Figma mockup of a user registration interface. At the top left is the acronym 'NBH'. A search bar with the placeholder 'Search ...' is positioned at the top center. On the top right are two icons: a bell and a user profile. The main title 'Add User' is centered above a breadcrumb navigation path: 'Home > User > Add User'. On the left sidebar, there are three items: 'Dashboard' (with a dashboard icon), 'Users' (with a users icon, highlighted in purple to indicate it's the active section), and another 'Users' item with a different icon. The main content area contains four input fields: 'Username' (placeholder 'Type name here ...'), 'Email' (placeholder 'Type email here ...'), 'Password' (placeholder 'Type Password here ...'), and 'Confirm Password' (placeholder 'Retype Password here ...'). At the bottom right of the form are two buttons: a blue 'Add' button and a grey 'Back' button.

Figure 66: Figma Mockup for registering new user.

3.7.1.2. Development

- Table Design

TABLE NAME			
users			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	sequence ◊ users_id_seq	PRIMARY KEY +
username	text	no default ◊	UNIQUE NOT NULL +
password	text	no default ◊	NOT NULL +
user_type	text	no default ◊	+
email	text	no default ◊	UNIQUE +
using_default_password	integer	expression ◊ 1	+

Indexes			
INDEX NAME	TYPE	COLUMNS	
users_pkey	Primary Key Index	id	
users_username_key	Unique Index	username	
users_email_key	Unique Index	email	

Figure 67: Table Design for user's table

All the above attributes were created in the table according to ERD developed. user_type however is only required in Sprint 5 but was included in the table.

- Frontend Development

The screenshot shows a user interface for adding a new user. At the top left is a logo with two blue vertical bars and the text "NBH". To its right is a search bar with placeholder text "Search...". On the far right are three small icons: a bell, a person, and a question mark. Below the header, there's a breadcrumb navigation: "Dashboard" (with a house icon), "Home / Users / Add User". A purple button labeled "+ User" is on the left. The main content area has a white background with a thin gray border. It contains several input fields:

- "User Name" with placeholder "Type name here".
- "Email" with placeholder "Type email here".
- "Password" with placeholder "Type password here".
- "Confirm Password" with placeholder "Retype password here".
- "User Type" with placeholder "Type user type here". This field is part of a dropdown menu indicated by a downward arrow.

At the bottom right of the form are two buttons: a purple "Add" button and a gray "Back" button. The entire form area is highlighted with a thick red border, and a red arrow points from the bottom left towards the "User Type" field.

Figure 68: Developed Frontend UI from mockup for adding users.

```

src > Pages > User > AddUser.jsx > [o] AddUser
  1 import React, { useRef, useState } from "react";
  2 import { useForm, Controller } from "react-hook-form";
  3 import axios from "axios";
  4 import { ToastContainer, toast } from "react-toastify";
  5 import "react-toastify/dist/ReactToastify.css";
  6 import { useNavigate, Link } from "react-router-dom";
  7 import Select from "react-select";
  8 import { useEffect } from "react";
  9 const AddUser = () => {
10   const navigate = useNavigate();
11   const notify = () => toast.success("User added successfully");
12   const notifyError = (msg) => toast.error(`$${msg}`);
13   const [renderApp, setRenderApp] = useState(false)
14   const [userType, setUserType] = useState([..])
15   const [selectedUserType, setSelectedUserType] = useState();
16   const [selectedRole, setSelectedRole] = useState([]);
17   const [permissions, setPermissions] = useState([]);
18   const [..] = useForm();
19   const password = useRef({});
20   password.current = watch('password', '');
21   const onSubmit = async (data) => {..}
22   };
23   const handleChildChange = (e, id) => {..}
24   const loadData = async () => {..}
25   useEffect(() => {
26     loadData()
27   }, []);
28   return (
29     <div>
30       {renderApp &&
31        <div className='px-7 font-secondary py-10'>
32          <ToastContainer className='text-sm text-grey' />
33          <div className='font-bold text-2xl'>Add User</div>
34          <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>..
35          </div>
36          <div className='mb-white mt-5 p-8 rounded-md border font-thin'>
37            <form onSubmit={handleSubmit(onSubmit)} action=''\>
38              <div className='flex flex-col mb-5'>..
39              <div className='flex flex-col mb-5'>..
40              <div className='grid grid-cols-2'>..
41              <div className='flex flex-col mb-5'>..
42              <div errors?.user_type?.type === "required" && <p className='text-red-600 font-main text-sm mt-1'>User Type is required</p>;
43                &&
44                <div className='''>..
45                </div>
46            </div>
47          </div>
48        </div>
49      </div>
50    );
51  };
52  
```

Figure 69: Screenshot of code developed for adding user.

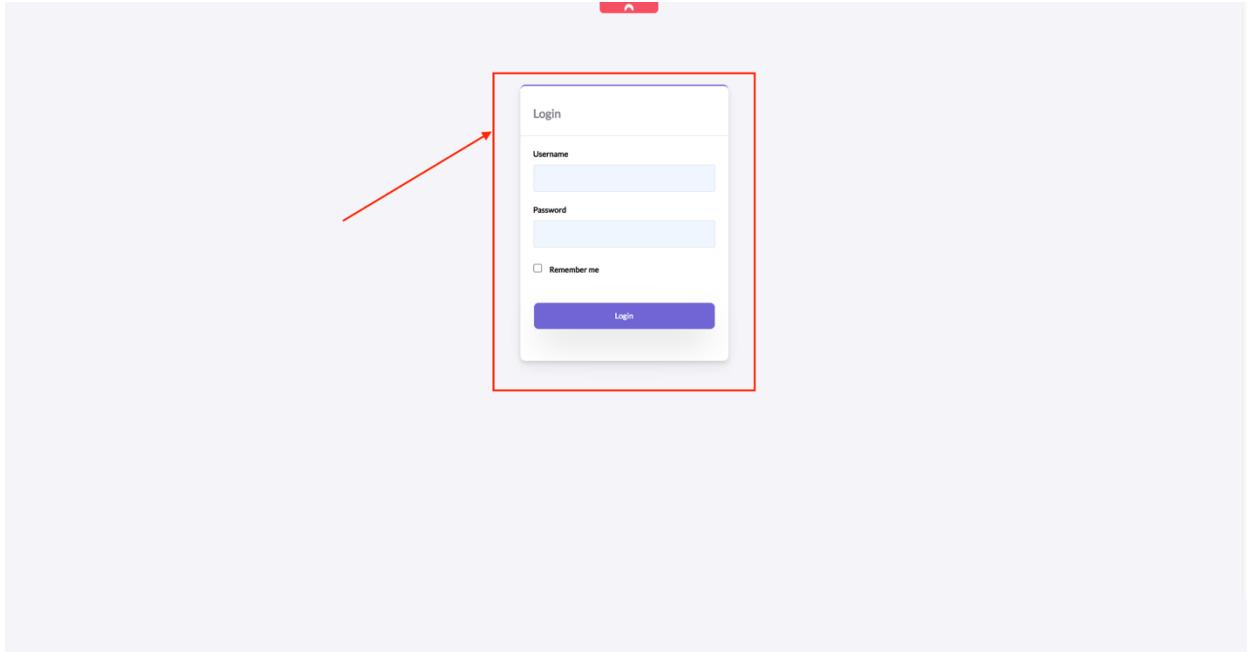


Figure 70: Developed Frontend UI from mockup for Login Page.

```

src > Pages > Login > Login.js > (e) Login > (e) onSubmit
  1 import React, { useState } from "react";
  2 import axios from "axios";
  3 import { useForm } from "react-hook-form";
  4 import { useNavigate } from "react-router-dom";
  5 import { useAuth } from "../../../Components/Authentication/auth";
  6 const Login = () => {
  7   const navigate = useNavigate();
  8   const auth = useAuth();
  9   const [success, setSuccess] = useState("");
10   const {
11     register,
12     handleSubmit,
13     setValue,
14     formState: { errors },
15   } = useForm();
16   const [errMsg, setErrMsg] = useState("");
17   const [loading, setLoading] = useState(false);
18   const [showPassword, setShowPassword] = useState(false)
19   const onSubmit = async (data) => {
20     setLoading(true);
21     const username = data.username;
22     const password = data.password;
23     try {
24       catch (err) {
25         setLoading(false);
26         setErrMsg(err.response?.data?.msg);
27         console.log(err);
28       }
29     };
30   return (
31     <div className='flex justify-center overflow-visible'>
32       <div className='border-t-primary rounded-lg drop-shadow-lg border-t-2 w-80 mt-32 bg-white flex justify-center'>
33         <div className='py-2 w-full'>
34           <div className='login-heading text-lg text-grey py-5 px-5 border-b'>
35             Login
36           </div>
37           <form
38             onSubmit={handleSubmit(onSubmit)}
39             className='px-5 pb-10'
40             action={property?.React.HTMLAttributes<T>.className?: string | undefined}
41             className?: string | undefined>-
42             <div className='row mt-5'>-
43             </div>
44             <div className='row mt-5'>-
45             </div>
46             <div className='row mt-10'>-
47             </div>
48           </form>
49         </div>
50       </div>
51     </div>
52   );
53 }
54
55
56
57
58
59
60
61
62 >
63 >
64 >
65 >
66 >
67 >
68 >
69 >
70 >
71 >
72 >
73 >
74 >
75 >
76 >
77 >
78 >
79 >
80 >
81 >
82 >
83 >
84 >
85 >
86 >
87 >
88 >
89 >
90 >
91 >
92 >
93 >
94 >
95 >
96 >
97 >
98 >
99 >
100 >
101 >
102 >
103 >
104 >
105 >
106 >
107 >
108 >
109 >
110 >
111 >
112 >
113 >
114 >
115 >
116 >
117 >
118 >
119 >
120 >
121 >
122 >
123 >
124 >
125 >
126 >
127 >
128 >
129 >
130 >
131 >
132 >
133 >
134 >
135 >
136 >
137 >
138 >
139 >
140 >
141 >
142 >
143 >
144 >
145 >
146 >
147 >
148 >
149 >
150 >
151 >
152 >
153 >
154 >
155 >
156 >
157 >
158 >
159 >
160 >
161 >
162 >
```

Figure 71: Screenshot of code developed for logging in.

3.7.1.3. Testing

a) API Testing

```
PASS __tests__/authentication.test.js
POST /api/register
  ✓ It should respond with missing parameter error (29 ms)
  ✓ It should respond with success message for registering a new user (65 ms)
POST /api/login
  ✓ It responds with success login response (59 ms)
  ✓ It responds with error login response (2 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.024 s
```

Figure 72: Screenshot of API testing result for Sprint 1

b) Cypress Testing

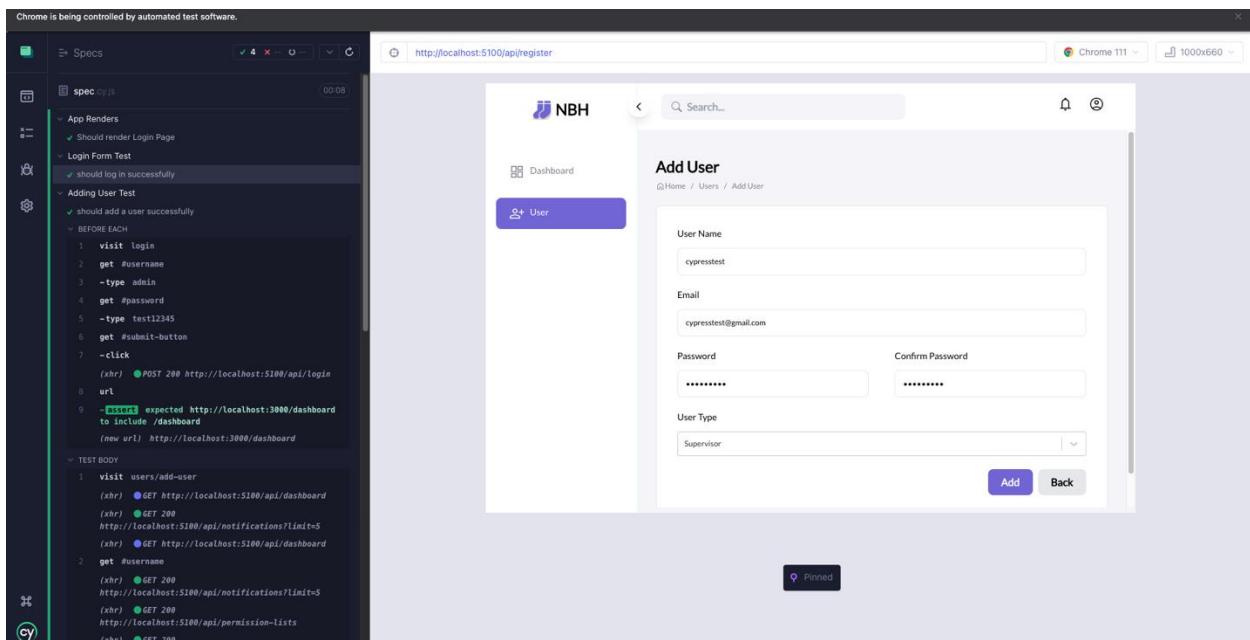


Figure 73: Screenshot of automatic testing result for react components.

3.7.1.4. Sprint Retrospective

In this retrospective report, I reviewed the outcomes of my sprint, which concentrated on developing user management features such as a login form and a super admin registration process. These features were effectively completed and tested for functionality. The frontend testing, however, was not finished due to a lack of understanding in automatic testing for react. In the future, I hope to add testing for both the frontend and the backend, as well as better the design of the login form.

3.7.2. Sprint 2 – Inventory Management System

3.7.2.1. Sprint Planning

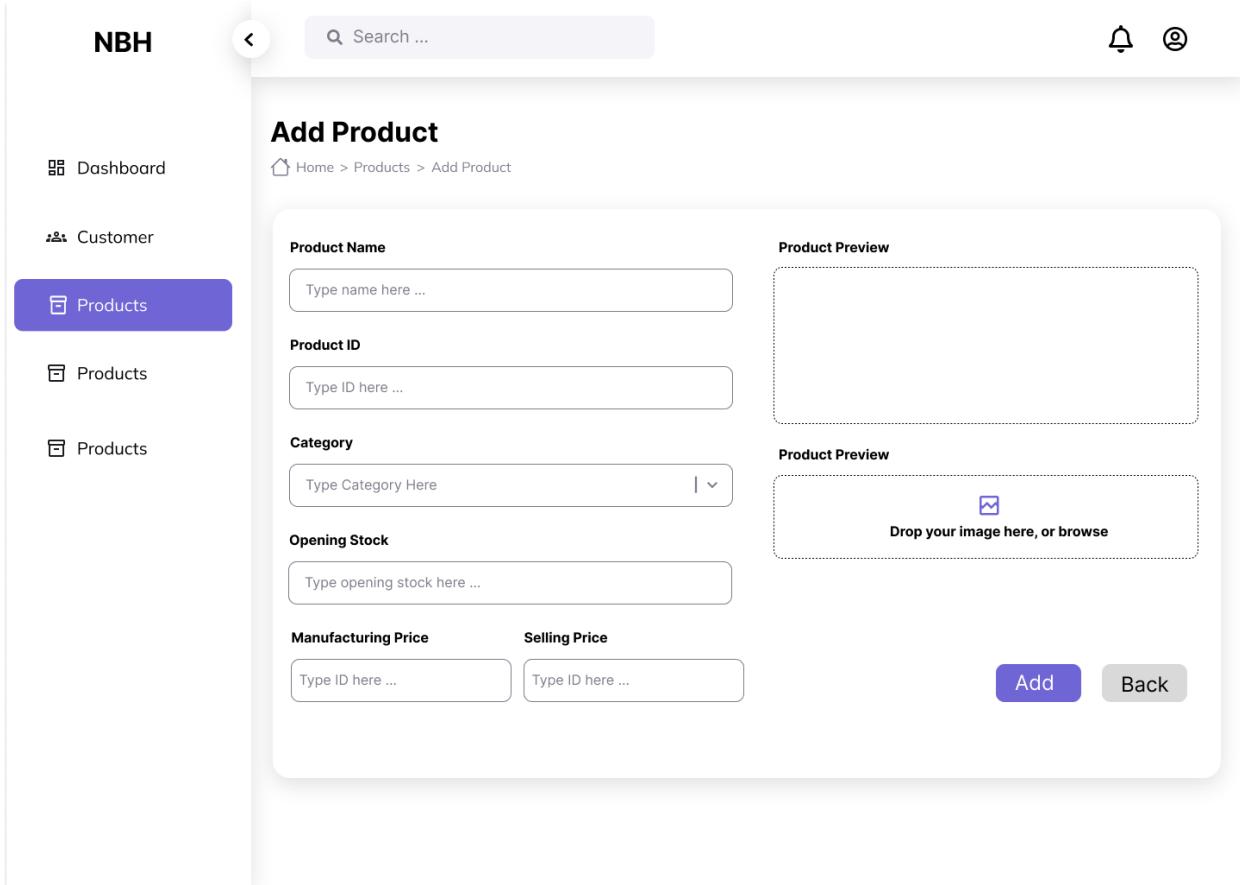
The second sprint of this project aims to build inventory management capabilities, which include developing CRUD operations for goods, designing mockups, and testing the features. The sprint's deliverables include fully working user administration capabilities, frontend, and backend components, and Figma mockups for adding new products and examining product data.

a) Figma Mockup

The figure shows a Figma mockup of a web-based inventory management system. The interface has a header with the logo 'NBH' and a search bar. On the left, there is a sidebar with navigation links: 'Dashboard', 'Customer', 'Products' (which is selected and highlighted in purple), and two other 'Products' links. The main content area is titled 'Products' and shows a list of items. The table columns are ID, Product Name, Current Stock, Image, Product Category, and Action. There are four items listed, all with ID P001 and P002, Product Name 'Fashion Socks', and Current Stock '0'. Each item has a small orange circular icon with a person icon next to it. In the 'Action' column, there are three dots and a context menu that includes 'View Product' and 'Edit Product'. At the bottom of the page, there is a pagination section showing 'Page 1 of 1 | Go to page: 1' and navigation buttons '<< Previous Next >>'.

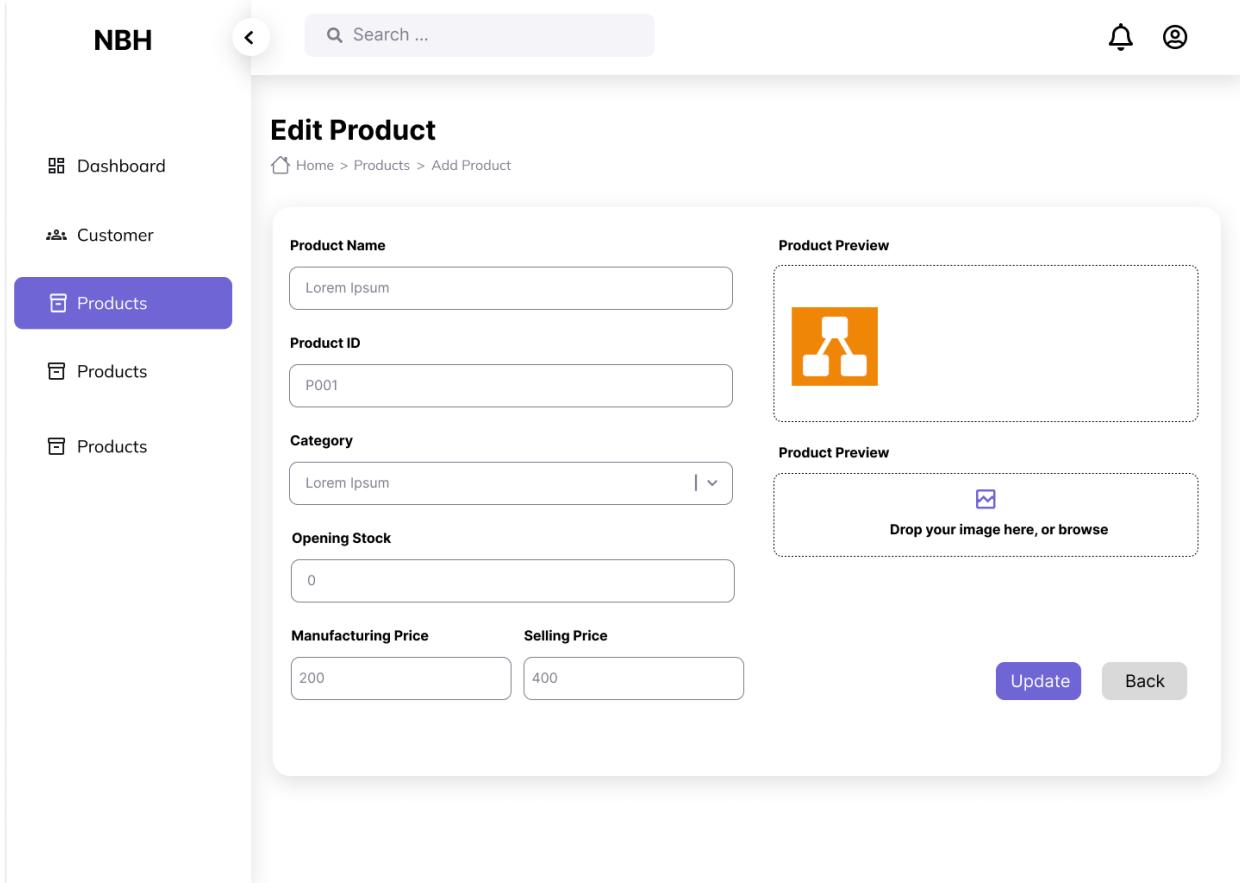
ID	Product Name	Current Stock	Image	Product Category	Action
P001	Fashion Socks	0		Product Category	⋮
P002	Fashion Socks	0		Product Category	⋮
P001	Fashion Socks	0		Product Category	⋮
P002	Fashion Socks	0		Product Category	⋮

Figure 74: Figma Mockup for Products Page



The image shows a Figma mockup of an 'Add Product' page. At the top left is a navigation bar with 'NBH' and a search bar. On the right are a bell icon and a user profile icon. A sidebar on the left has 'Dashboard', 'Customer', and several 'Products' options, with the third one highlighted in purple. The main content area has a title 'Add Product' and a breadcrumb 'Home > Products > Add Product'. It contains fields for 'Product Name', 'Product ID', 'Category' (with a dropdown arrow), 'Opening Stock', 'Manufacturing Price', and 'Selling Price'. To the right are two 'Product Preview' sections with dashed outlines, one for the product image and another for a preview of the details. A large button at the bottom right says 'Add'.

Figure 75: Figma Mockup for Add Product Page



The image shows a Figma mockup of an 'Edit Product' page. At the top left is a sidebar with a 'NBH' logo and navigation links: Dashboard, Customer, Products (which is highlighted in purple), Products, and Products. A search bar at the top right contains the placeholder 'Search ...'. The main content area has a title 'Edit Product' and a breadcrumb trail 'Home > Products > Add Product'. The form fields include 'Product Name' (Lorem Ipsum), 'Product ID' (P001), 'Category' (Lorem Ipsum), 'Opening Stock' (0), 'Manufacturing Price' (200), and 'Selling Price' (400). There are two 'Product Preview' sections: one showing a small orange icon of a thumbs-up and another with a dashed border and a 'Drop your image here, or browse' placeholder. Buttons for 'Update' and 'Back' are located at the bottom right.

Figure 76: Figma Mockup for Edit Product Page

3.7.2.2. Development

a) Table Design

TABLE NAME			
products			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	sequence ◊ products_product_id_seq	PRIMARY KEY +
name	text	no default ◊	UNIQUE +
price	smallint	no default ◊	NOT NULL +
opening_stock	integer	no default ◊	+
image	text	no default ◊	+
product_category	integer	no default ◊	→ product_categories.id +
product_id	text	no default ◊	UNIQUE NOT NULL +
cost_price	smallint	no default ◊	+
minimum_stock	integer	expression ◊ 0	+

Indexes		
INDEX NAME	TYPE	COLUMNS
products_pkey	Primary Key Index	id
products_name_key	Unique Index	name
products_product_id_key	Unique Index	product_id

Figure 77: Table Design for product's table

TABLE NAME			
product_categories			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	sequence ◊ product_categories_id_seq	PRIMARY KEY +
name	text	no default ◊	+
Indexes			
INDEX NAME	TYPE	COLUMNS	
product_categories_pkey	Primary Key Index	id	

Figure 78: Table Design for product category table

b) Frontend Development

ID	Product Name	Current Stock	Image	Product Category	Action
P003	Emoji Socks	-134		Half Socks	⋮
P008	Pride Socks	220		Ankle Socks	⋮
P001	Fashion Socks	1361		Ankle Socks	⋮
P004	Kappa Socks	68		Ankle Socks	⋮
P002	Ladies Ankle Sock	1446		Ankle Socks	⋮
P007	Baby Socks	400		Ankle Socks	⋮
P013	Jordan Socks	268		Half Socks	⋮
P005	New Products	911		Half Socks	⋮
P014	Test Product	0		Ankle Socks	⋮
P012	Game Sock	2492		Ankle Socks	⋮

Page 1 of 1 | Go to page: Total: 10

<< Previous Next >>

Figure 79: Developed Frontend UI from mockup for viewing product list.

```

src > Pages > Products > Product.js ...
1  import React, { useEffect, useState } from "react";
2  import { Link, useLocation } from "react-router-dom";
3  import { Table } from "./Product-Table/Table";
4  import axios from "axios";
5
6  import { ToastContainer, toast } from "react-toastify";
7  import "react-toastify/dist/ReactToastify.css";
8  import { useAuth } from "../../Components/Authentication/auth";
9  const Product = () => {
10    const auth = useAuth();
11    const location = useLocation();
12    const [allProducts, setAllProducts] = useState([]);
13    const [renderApp, setRenderApp] = useState(false);
14    const [filterData, setFilterData] = useState([]);
15    const [category, setCategory] = useState();
16    > const loadData = async () => {
39    };
40    > useEffect(() => {
41      [location];
44
45    > const notify = () => {...}
48  );
49  const notifyError = (msg) => toast.error(`$${msg}`);
50  > const handleDelete = async (data) => {...}
52  );
63
64  > const handleChange = async (value) => {...}
70
71
72  const handleEdit = async (data) => {};
73
74  return (
75    <div className='px-7 font-secondary py-10'>
76      <ToastContainer className='text-sm text-grey' />
77      <div className='flex justify-between'>
78        <div className='>
79          <div className='font-bold text-2xl'>Products</div>
80        >
81          <div (property React.HTMLAttributes<HTMLDivElement>.className?: string | undefined | grey'>...
82            <d className?: string | undefined;
83          </d>
84          <div className='>...
85            </div>
86          </div>
87        </div>
88        <div className='product-table'>
89          <Table handleDelete={handleDelete} handleChange={handleChange} category={category} data={filterData} loadData={loadData} />
90        </div>
91      </div>
92    );
93  );
94
95  export default Product;

```

Figure 80: Screenshot of code developed for viewing product list.

The screenshot shows a "Add Product" form. At the top left is a breadcrumb navigation: Home / Products / Add Product. The form has several input fields: "Product Name" (placeholder: Type name here), "Product ID" (placeholder: Type ID here), "Category" (dropdown placeholder: Type Category here), "Opening Stock" (placeholder: Type opening stock here), "Manufacturing Price" (placeholder: Type manufacturing price here), and "Selling Price" (placeholder: Type opening stock here). To the right of these fields is a "Product Preview" section with a dashed border, a "Product Gallery" section with a file input field ("Drop your image here or browse"), and two buttons at the bottom right: "Add" (purple) and "Back" (grey).

Figure 81: Developed Frontend UI from mockup for adding product.

```

src > Pages > Products > AddProduct.js ...
1  import React, { useEffect, useState } from "react";
2  import Select from "react-select";
3  import { useForm, Controller } from "react-hook-form";
4  import axios from "axios";
5  import { ToastContainer, toast } from "react-toastify";
6  import "react-toastify/dist/ReactToastify.css";
7  import { useNavigate, Link } from "react-router-dom";
8  const AddProduct = () => {
9    const navigate = useNavigate();
10   const notify = () => toast.success("Product added successfully");
11   const notifyError = (msg) => toast.error(`${msg}`);
12   ...
13   } = useForm();
14   const [categoryOption, setCategoryOption] = useState([]);
15   const [taxOption, setTaxOption] = useState([]);
16   const [renderApp, setRenderApp] = useState(false);
17
18   const [uploadedImage, setUploadedImage] = useState(null);
19   const imageChange = (e) => {
20     setUploadedImage(URL.createObjectURL(e.target.files[0]));
21   };
22   const handleClick = () => {
23     document.getElementById("image-holder").click();
24   };
25   const onSubmit = async (data) => {
26     ...
27   };
28   const removeSelectedImage = () => {
29     setUploadedImage();
30   };
31
32   const loadData = async () => {
33   };
34   useEffect(() => {
35     loadData();
36   }, []);
37   return (
38     <div className='px-7 font-secondary py-10'>
39       {renderApp && (
40         <div>
41           <div>Add Product</div>
42           <ToastContainer className='text-sm text-grey' />
43           <div className='breadrum-container flex items-center mt-1 text-xs font-thin text-grey'>
44             <div>Products</div>
45             <div>Add Product</div>
46           </div>
47           <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
48             <form onSubmit={handleSubmit(onSubmit)} action='>...
49             </form>
50           </div>
51         </div>
52       </div>
53     </div>
54   );
55 }

```

Figure 82: Screenshot of code developed for adding new product.

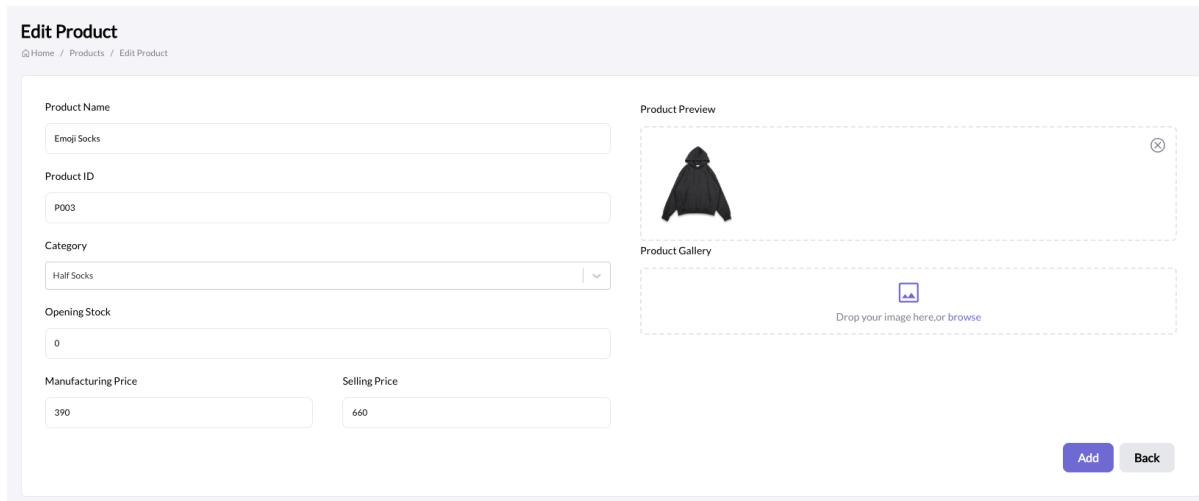


Figure 83: Developed Frontend UI from mockup for editing product.

```

src > Pages > Products > ↗ EditProduct.js > ↗ EditProduct
  1 import React, { useEffect, useState } from "react";
  2 import Select from "react-select";
  3 import { useForm, Controller } from "react-hook-form";
  4 import axios from "axios";
  5 import { ToastContainer, toast } from "react-toastify";
  6 import "react-toastify/dist/ReactToastify.css";
  7 import { useNavigate, Link, useParams } from "react-router-dom";
  8 const EditProduct = () => [
  9   const { id } = useParams();
 10   const [dbData, setDbData] = useState([]);
 11   const navigate = useNavigate();
 12   const notify = () => toast.success("Product updated successfully");
 13   const notifyError = (msg) => toast.error(`$${msg}`);
 14   const [
 15     ...useForm(),
 16     { categoryOption, setCategoryOption } = useState([1]),
 17   ];
 18   const [renderApp, setRenderApp] = useState(false);
 19   const [uploadedImage, setUploadedImage] = useState(null);
 20   const imageChange = (e) => {
 21     setRenderApp(true);
 22   };
 23   const handleClick = () => {
 24     setRenderApp(false);
 25   };
 26   const onSubmit = async (data) => {
 27     try {
 28       await axios.put(`http://localhost:5000/api/products/${id}`, data);
 29       navigate(`/products`);
 30     } catch (err) {
 31       notifyError(err.message);
 32     }
 33   };
 34   const removeSelectedImage = () => {
 35     setUploadedImage(null);
 36   };
 37   const loadData = async () => {
 38     try {
 39       const res = await axios.get(`http://localhost:5000/api/products/${id}`);
 40       setDbData(res.data);
 41     } catch (err) {
 42       notifyError(err.message);
 43     }
 44   };
 45   useEffect(() => {
 46     loadData();
 47   }, []);
 48   return (
 49     <div className='px-7 font-secondary py-10'>
 50       {renderApp && (
 51         <>
 52           <div className='font-bold text-2xl'>Edit Product</div>
 53           <ToastContainer className='text-sm text-grey' />
 54           <div className='breadrum-container flex items-center mt-1 text-xs font-thin text-grey'>...
 55           </div>
 56           <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
 57             <form onSubmit={handleSubmit(onSubmit)} action='-'>...
 58               <input type='text' name='name' value={dbData.name} />
 59               <input type='text' name='category' value={categoryOption.value} />
 60               <input type='text' name='stock' value={dbData.stock} />
 61               <input type='text' name='manufacturingPrice' value={dbData.manufacturingPrice} />
 62               <input type='text' name='sellingPrice' value={dbData.sellingPrice} />
 63             </form>
 64           </div>
 65         </>
 66       )
 67     </div>
 68   );
 69   export default EditProduct;

```

Figure 84: Screenshot of code developed for editing product.

3.7.2.3. Testing

a) API Testing

```
> test
> jest --runInBand --force-exit

PASS  __tests__/product.test.js
PASS  __tests__/authentication.test.js

Test Suites: 2 passed, 2 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        1.902 s, estimated 2 s
Ran all test suites.
```

Figure 85: Screenshot of API testing result for Sprint 2

b) Cypress Testing

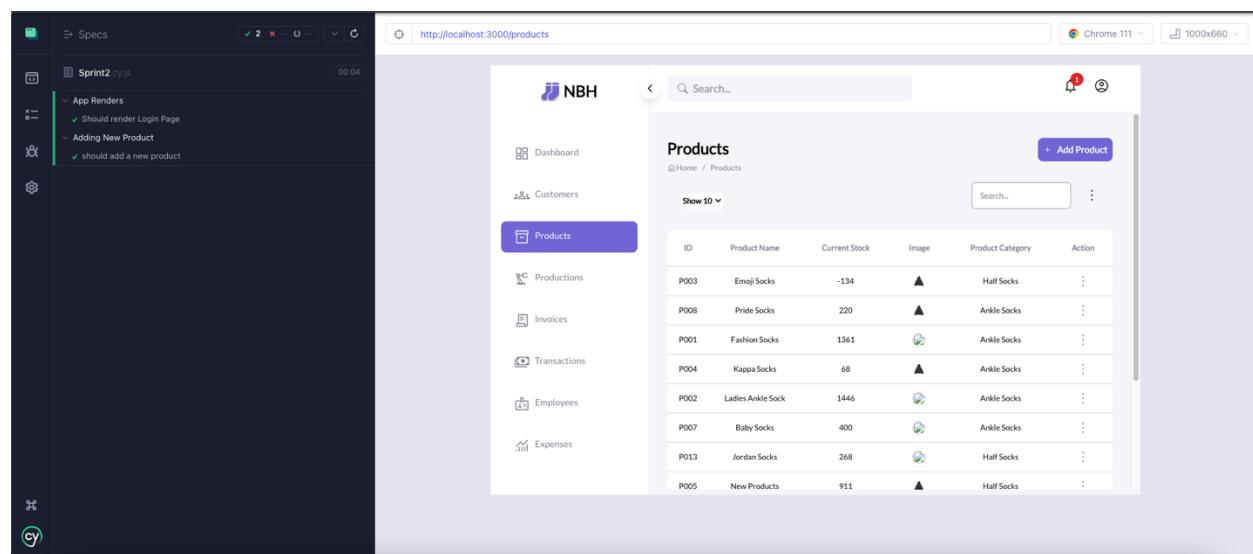


Figure 86: Screenshot of automatic test result for react components.

3.7.2.4. Sprint Retrospective

The sprint goal was to add products to the database and test inventory management for the backend, which was achieved successfully. However, the frontend testing was not completed due to a lack of understanding in automatic testing for React. To improve the sprint in the future, the team plans to review coding style, break down tasks, and learn more about automatic testing for React.

3.7.3. Sprint 3 – Invoice Billing System

3.7.3.1. Sprint Planning

The sprint objective is to develop an invoice billing system that enables CRUD operations on products and tracks inventory changes. The tasks involve creating a functional system, Figma mockups, database table based on the ERD, and performing CRUD operations for customers. The deliverables are a tested and functional billing system with both frontend and backend components, along with Figma mockups for adding new invoices and customers.

a) Figma Mockup

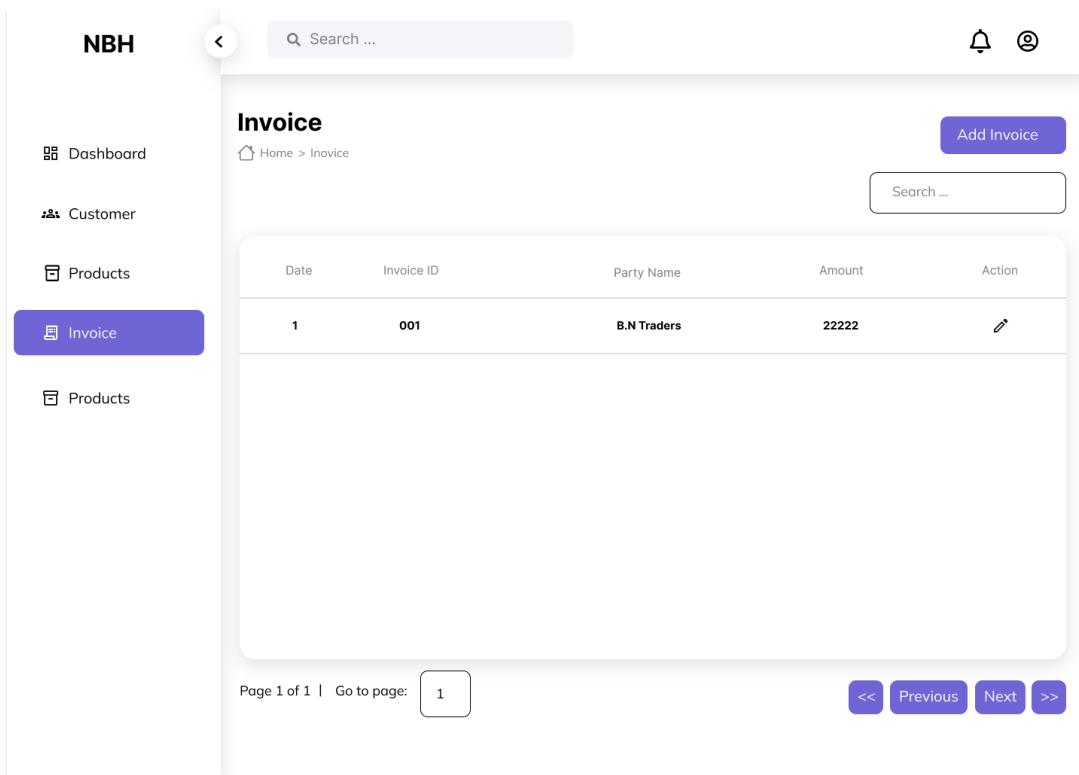
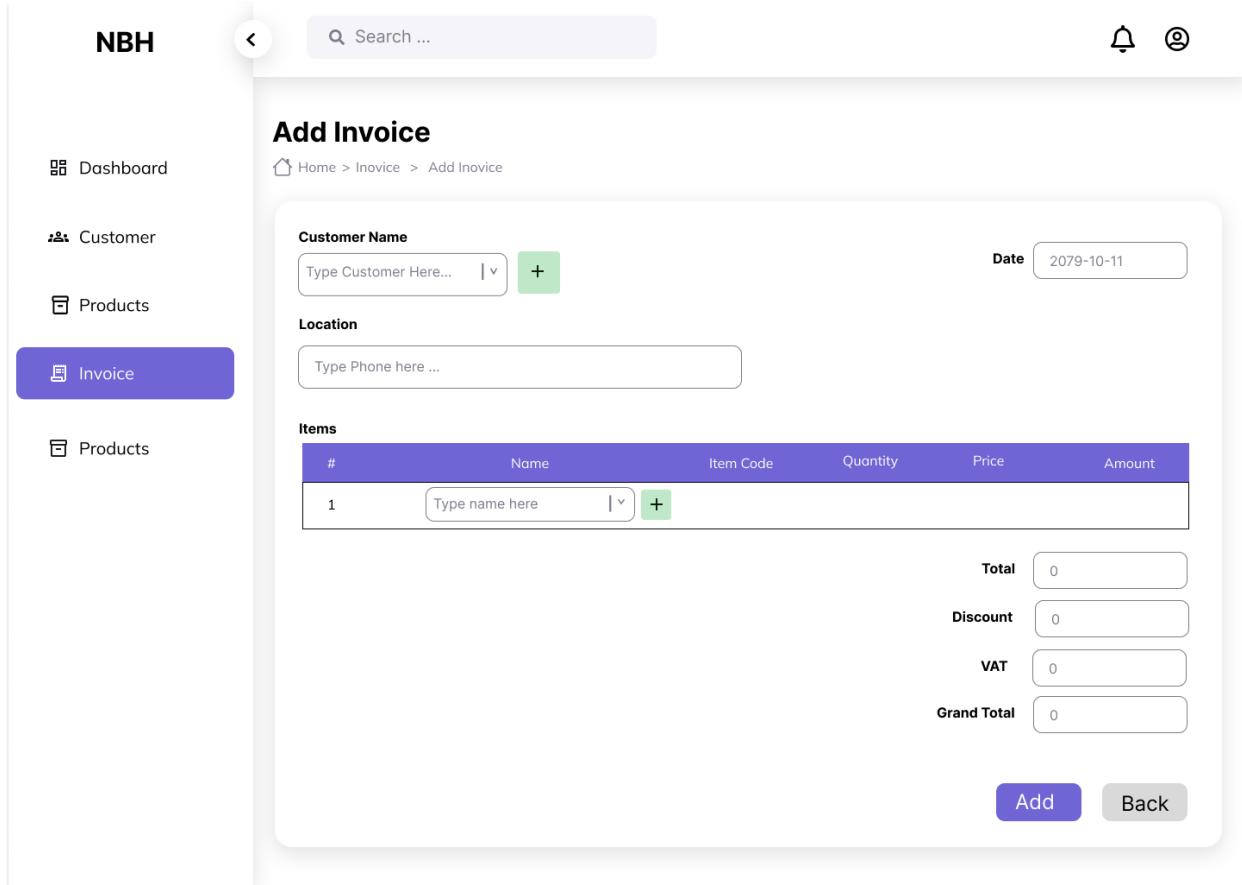


Figure 87: Figma Mockup for Invoice Page



The image shows a Figma mockup of an 'Add Invoice' page. At the top left is the logo 'NBH'. A search bar with placeholder text 'Search ...' is at the top right, along with a bell icon and a user profile icon.

The main title 'Add Invoice' is centered above the form fields. Below it is a breadcrumb navigation: Home > Inovice > Add Inovice.

The form fields include:

- Customer Name:** An input field with placeholder 'Type Customer Here...' and a green '+' button.
- Date:** A date picker set to '2079-10-11'.
- Location:** An input field with placeholder 'Type Phone here ...'.
- Items:** A table with columns: #, Name, Item Code, Quantity, Price, and Amount. It has one row with '# 1' and 'Name' placeholder 'Type name here'.
- Calculations:** Fields for Total (0), Discount (0), VAT (0), and Grand Total (0).

At the bottom are two buttons: 'Add' (purple) and 'Back' (gray).

Figure 88: Figma Mockup for Add Invoice Page

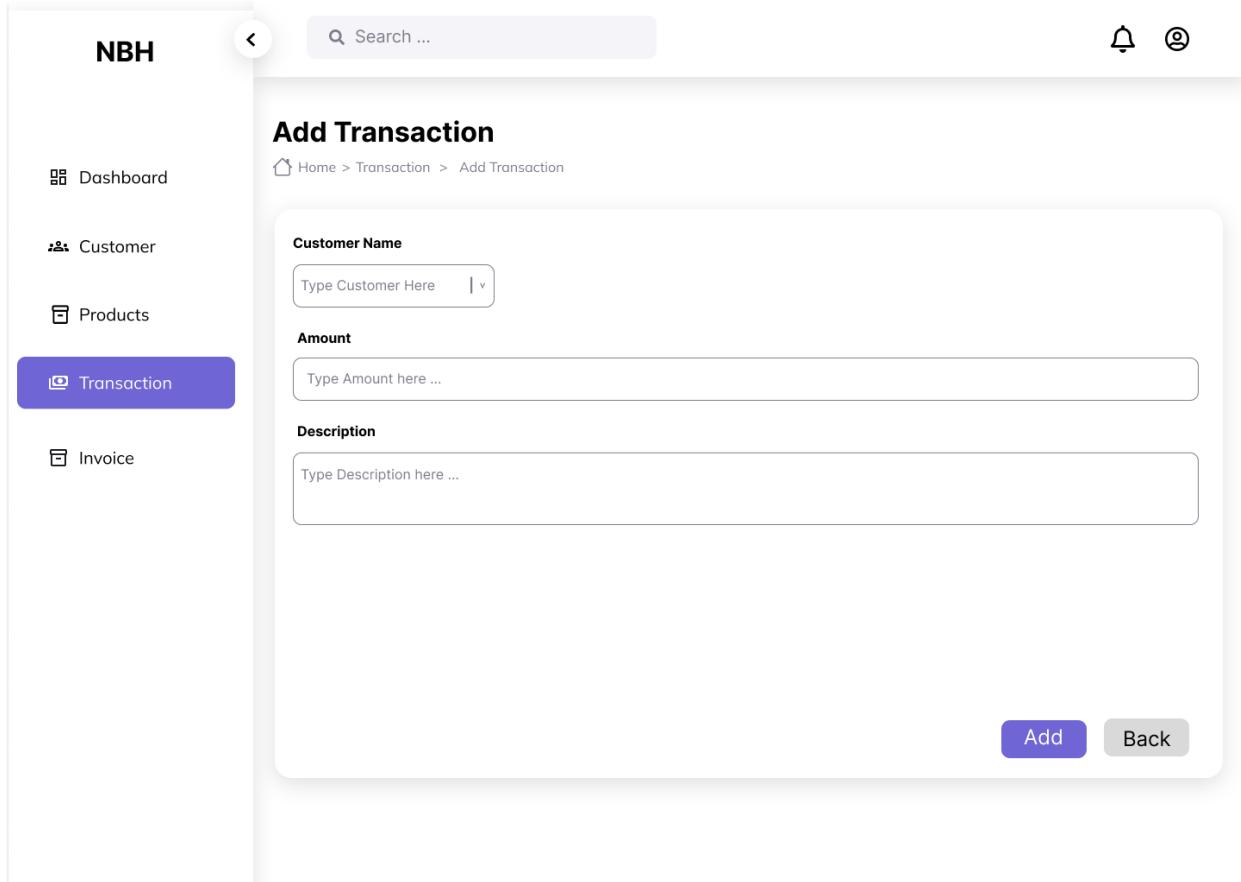
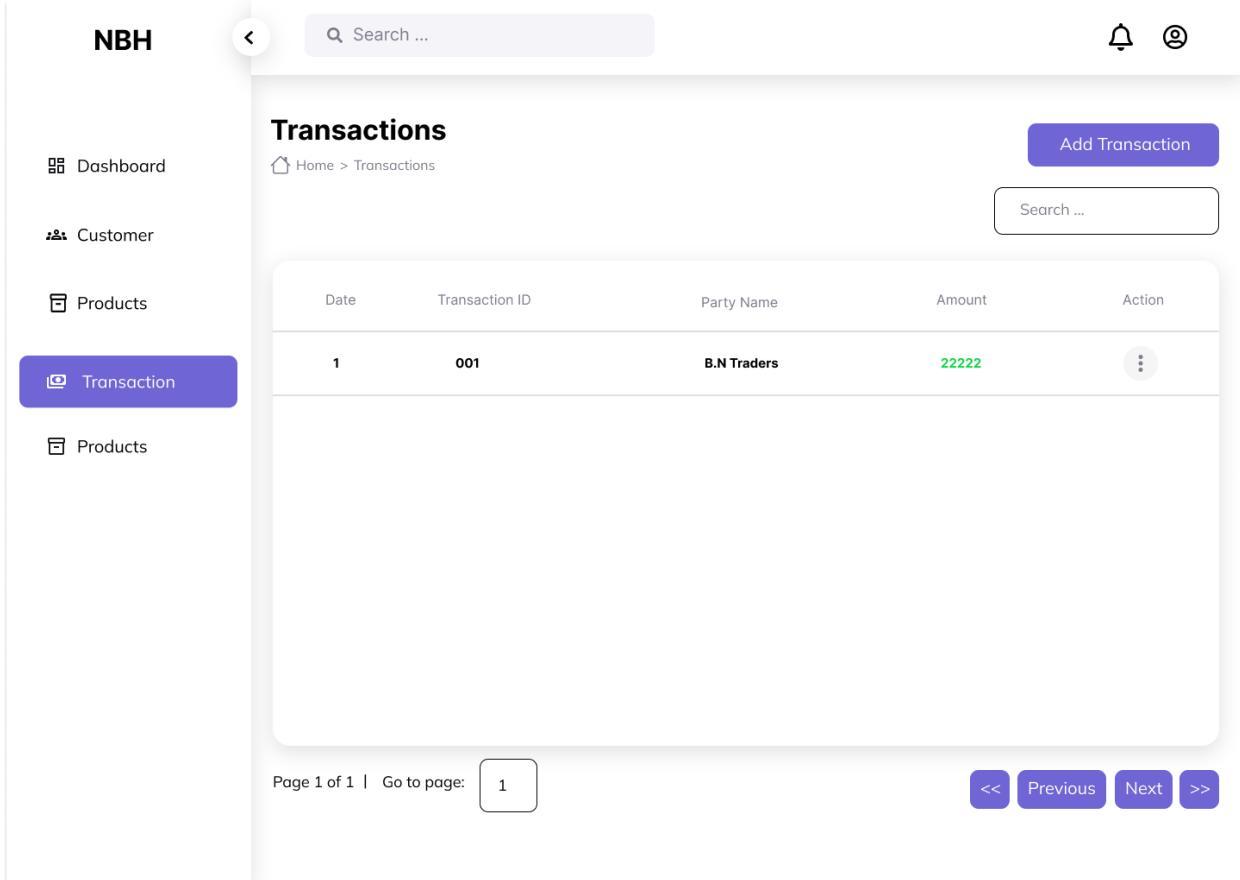


Figure 89: Figma Mockup for Add Transaction Page



The image shows a Figma mockup of a transaction page. At the top left is a sidebar with the title "NBH". The sidebar contains several menu items: "Dashboard", "Customer", "Products", and "Transaction". The "Transaction" item is highlighted with a purple background. At the top right are icons for search, notifications, and user profile. The main content area has a header "Transactions" with a breadcrumb "Home > Transactions" and a "Add Transaction" button. Below is a table with one row of data:

Date	Transaction ID	Party Name	Amount	Action
1	001	B.N Traders	2222	⋮

At the bottom, there is a footer with pagination controls: "Page 1 of 1 | Go to page: 1" and buttons for "Previous", "Next", and "Last".

Figure 90: Figma Mockup for Transaction Page

The Figma mockup displays a customer profile for 'B.N Traders'. On the left, a sidebar shows navigation links: Dashboard, Customer (highlighted in purple), Products, Products, and Products. The main content area has a header with a back arrow, the title 'B.N Traders', and a search bar. Below the header, a breadcrumb trail shows 'Home > Customers > B.N Traders'. Three summary boxes show 'Total Sales Rs. 2200' (Total for 365 days), 'Remaining Balance Rs. 2200' (Total for 365 days), and 'Customer Information' with details: Name (B.N Traders), Phone (015329952), Email (bhavuknepal83@gmail.com), Location (New Road, Kathmandu), and PAN (604326669). A 'Recent Purchase' section is shown below.

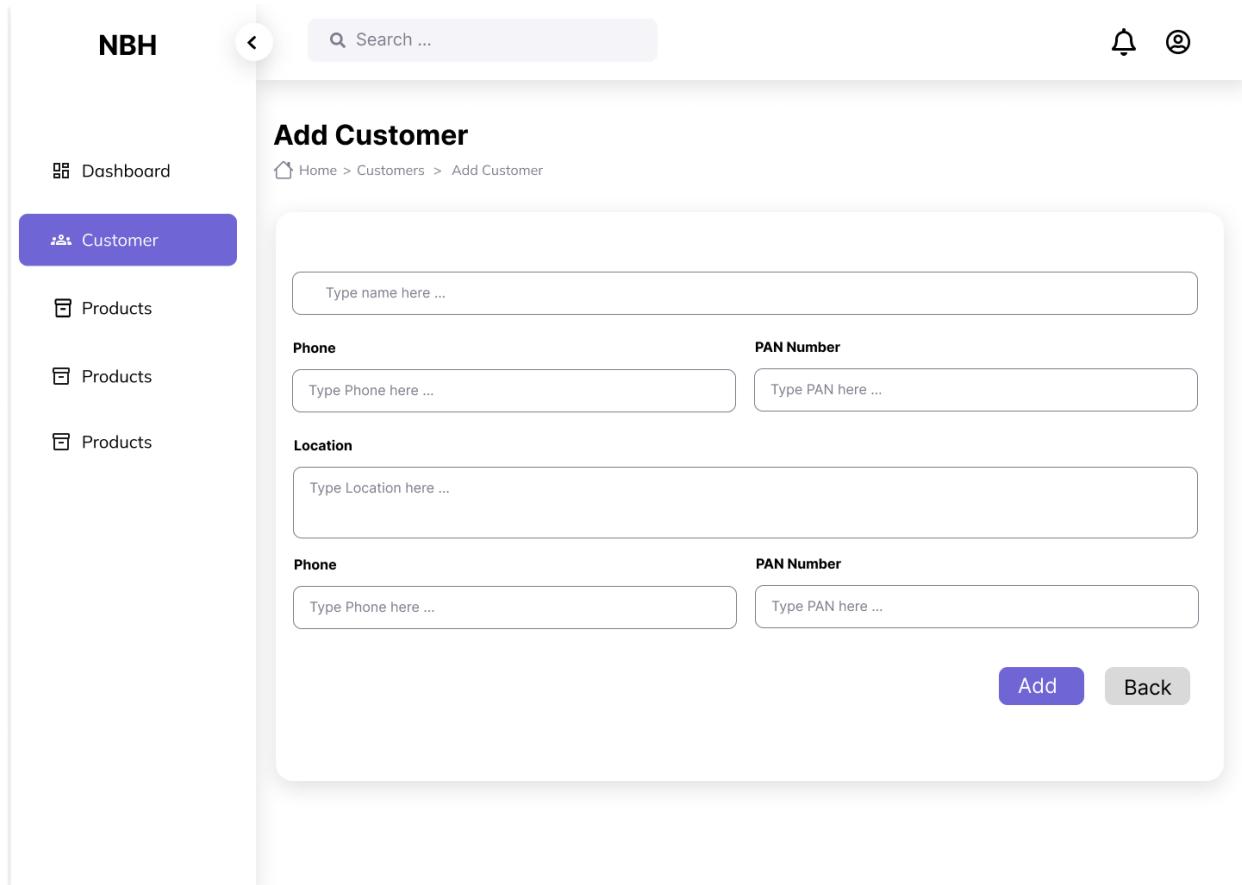
Figure 91: Figma Mockup for Customer Information Page

The image shows a Figma mockup of a web application interface. On the left is a sidebar with the title "NBH". It contains several navigation items: "Dashboard" (with a bar chart icon), "Customer" (selected, indicated by a purple background), and three "Products" items (each with a bar chart icon). At the top right of the main content area are a search bar ("Search ..."), a bell icon, and a user profile icon. The main content area has a header "Customers" and a breadcrumb "Home > Customers". A purple button "Add Customer" is located in the top right corner of the content area. Below the header is a search bar labeled "Search ...". The main content is a table with the following data:

ID	Name	Location	Phone	Balance	Status	Action
1	B.N Traders	Newroad,Kathmandu	015329953	-22222	Active	⋮
2	Singh General Store	Dhangadi,Nepal	98179898988	400000	Active	⋮

At the bottom of the content area, there is a footer with the text "Page 1 of 1 | Go to page: 1" and navigation buttons: "<<" (disabled), "Previous" (disabled), "Next", and ">>".

Figure 92: Figma Mockup for Customer Page



The image shows a Figma mockup of an 'Add Customer' page. At the top left is the logo 'NBH'. A search bar with placeholder text 'Search ...' is positioned at the top center. On the top right are a bell icon and a user profile icon. The main title 'Add Customer' is centered above a breadcrumb navigation path: 'Home > Customers > Add Customer'. To the left of the form is a sidebar with a dark blue header containing 'Customer' and several other menu items: 'Dashboard', 'Products', 'Products', 'Products', and 'Products'. The main form area contains two sections for customer information. The first section has a 'Name' input field labeled 'Type name here ...'. Below it are two side-by-side input fields for 'Phone' and 'PAN Number', both with placeholder text 'Type Phone here ...' and 'Type PAN here ...' respectively. The second section has a 'Location' input field labeled 'Type Location here ...'. Below it are two more side-by-side input fields for 'Phone' and 'PAN Number', both with placeholder text 'Type Phone here ...' and 'Type PAN here ...' respectively. At the bottom right of the form are two buttons: a purple 'Add' button and a grey 'Back' button.

Figure 93: Figma Mockup for Add Customer Page

3.7.3.2. Development

a) Table Design

TABLE NAME			
<code>customers</code>			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
<code>id</code>	integer	identity	PRIMARY KEY +
<code>name</code>	text	no default	NOT NULL +
<code>phone</code>	text	no default	NOT NULL +
<code>email</code>	text	no default	+ (highlighted)
<code>location</code>	text	no default	NOT NULL +
<code>balance</code>	numeric(12,2)	no default	+
<code>pan</code>	integer	no default	UNIQUE NOT NULL +
<code>status</code>	text	constant active	+
<code>district</code>	integer	no default	→ nepal_districts.id +

Indexes		
INDEX NAME	TYPE	COLUMNS
<code>customers_pkey</code>	Primary Key Index	<code>id</code>
<code>customers_pan_key</code>	Unique Index	<code>pan</code>

Figure 94: Table Design for customer's table

TABLE NAME			
<code>nepal_districts</code>			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
<code>id</code>	integer	identity ◊	PRIMARY KEY +
<code>name</code>	text	no default ◊	+
Indexes			
INDEX NAME	TYPE	COLUMNS	
<code>nepal_districts_pkey</code>	Primary Key Index	<code>id</code>	

Figure 95: Table Design for Nepal's district table

TABLE NAME			
<code>money_transaction</code>			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
<code>id</code>	integer	identity ◊	PRIMARY KEY +
<code>customer</code>	integer	no default ◊	→ customers.id +
<code>amount</code>	numeric(10,2)	no default ◊	amount > 0::numeric +
<code>description</code>	text	no default ◊	+
<code>date</code>	date	no default ◊	+
<code>transaction_id</code>	text	no default ◊	UNIQUE +
Indexes			
INDEX NAME	TYPE	COLUMNS	
<code>money_transaction_pkey</code>	Primary Key Index	<code>id</code>	
<code>money_transaction_transaction_id_key</code>	Unique Index	<code>transaction_id</code>	

Figure 96: Table Design for money transaction table

TABLE NAME			
invoices			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity ◊	PRIMARY KEY +
date	date	no default ◊	NOT NULL +
customer	integer	no default ◊	→ customers.id NOT NULL +
sub_total	integer	no default ◊	+
discount	numeric(12,2)	no default ◊	+
vat	numeric(11,2)	no default ◊	+
grand_total	numeric(12,2)	no default ◊	+

Indexes			
INDEX NAME	TYPE	COLUMNS	
invoices_pkey	Primary Key Index	id	

Figure 97: Table Design for invoices table

TABLE NAME			
invoice_products			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity ◊	PRIMARY KEY +
invoice	integer	no default ◊	→ invoices.id +
product	integer	no default ◊	→ products.id +
quantity	integer	no default ◊	NOT NULL +
price	integer	no default ◊	+
amount	integer	no default ◊	+

Indexes			
INDEX NAME	TYPE	COLUMNS	
invoice_products_pkey	Primary Key Index	id	

Figure 98: Table Design for invoices product table

TABLE NAME			
product_transcation			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity	PRIMARY KEY +
product	integer	no default	→ products.id +
quantity	integer	no default	+
transcation	text	no default	+
description	text	no default	+
transaction_date	date	no default	NOT NULL +

Indexes		
INDEX NAME	TYPE	COLUMNS
product_transcation_pkey	Primary Key Index	id

Figure 99: Table Design for product transaction table

b) Frontend Development

Date	Invoice ID	Party Name	Amount	Action
2079/12/08	48	Lumbini Fancy Store	67800.00	⋮
2079/12/12	49	B.N Traders	76162.00	⋮
2079/12/19	50	B.N Traders	41810.00	⋮
2079/11/04	27	Singh General Store	59664.00	⋮
2079/10/03	28	Singh General Store	162720.00	⋮
2079/11/04	29	B.N Traders	112322.00	⋮
2079/11/04	30	AB Store	24521.00	⋮
2079/11/05	31	H Tigers Footware	58760.00	⋮
2079/11/08	32	H Tigers Footware	363069.00	⋮
2079/11/10	33	Singh General Store	61020.00	⋮

Page 1 of 3 | Go to page: 1 Total: 24

Figure 100: Screenshot of developed invoice page

```

src > Pages > Invoice > Invoice.js > Invoice
  1 import React, { useEffect, useState } from "react";
  2 import { Link, useLocation } from "react-router-dom";
  3 import { Table } from "./Invoice-Table/Table";
  4 import axios from "axios";
  5
  6 import { ToastContainer, toast } from "react-toastify";
  7 import "react-toastify/dist/ReactToastify.css";
  8 import { useAuth } from "../../Components/Authentication/auth";
  9 const Invoice = () => {
10   const location = useLocation();
11   const [allInvoice, setAllInvoices] = useState([]);
12   const [renderApp, setRenderApp] = useState(false);
13   const auth = useAuth();
14   const loadData = async () => {
15   };
16   useEffect(() => {
17     loadData();
18   }, [location]);
19
20   return (
21     <div className='px-7 font-secondary py-10'>
22       <ToastContainer className='text-sm text-grey' />
23       <div className='flex justify-between'>
24         <div className='flex items-center'>
25           <div className='font-bold text-2xl'>Invoice</div>
26           <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
27             <div className='first flex items-center'>
28               <span className='material-symbols-outlined text-sm'>home</span>
29               <span className='!>Home</span>
30             </div>
31             <div className='divider mx-2'></div>
32             <div className='second'>Invoice</div>
33           </div>
34         </div>
35         <div className='flex items-center'>
36           <Link
37             to='/billing/add-billing'
38             className='bg-primary text-white p-2 text-sm flex items-center font-main rounded-lg'>
39             <span className='material-symbols-outlined text-sm mr-2'>add</span>
40             Add Invoice
41           </Link>
42         </div>
43       </div>
44     </div>
45   );
46   <div className='Invoice-table'>
47     {renderApp && <Table data={allInvoice} />}
48   </div>
49 };
50
51
52
53
54
55
56
57
58
59 export default Invoice;
60

```

Figure 101: Screenshot of code developed for viewing invoices list.

The screenshot shows a web-based application for adding an invoice. At the top left is the title "Add Invoice". Below it is a breadcrumb navigation: Home / Invoice / Add Invoice. The main form area contains fields for "Customer Name" (with a dropdown menu and a green "+" button) and "Location" (with a text input field). Below these is a table titled "Items" with columns: #, Name, Item code, Quantity, Price, and Amount. A single row is present with an ID of 1. At the bottom right of the form, there is a "Total:" field showing "0" and two buttons: "Add" and "Back".

Figure 102: Screenshot of developed add invoice page.

```

src > Pages > Invoice > AddInvoice.js ...
1 import React, { useState, useEffect, useCallback } from "react";
2 import { useForm, Controller } from "react-hook-form";
3 import Select from "react-select";
4 import axios from "axios";
5 import { Link, useParams, useNavigate } from "react-router-dom";
6 import AddCustomer from "../../Components/AddCustomer/AddCustomer";
7 import AddProduct from "../../Components/AddProduct/AddProduct";
8 import { ToastContainer, toast } from "react-toastify";
9 import "react-toastify/dist/ReactToastify.css";
10 import { adToBs } from "@sbmdkl/nepali-date-converter";
11 import format from "date-fns/format";
12 import Calendar from "@sbmdkl/nepali-datepicker-reactjs";
13 import "@sbmdkl/nepali-datepicker-reactjs/dist/index.css";
14 import QrCodeReader from "react-qrcode-reader";
15 import { useAuth } from "../../Components/Authentication/auth";
16
17
18 const AddInvoice = () => {
19   const { id } = useParams();
20   const [customerView, setCustomerView] = useState(false);
21   const [productView, setProductView] = useState(false);
22   const [renderApp, setRenderApp] = useState(false);
23   const [customerList, setCustomerList] = useState([]);
24   const [subTotal, setSubTotal] = useState(0);
25   const [discount, setDiscount] = useState(0);
26   const [discountValue, setDiscountValue] = useState(0);
27   const [vatAmount, setVatAmount] = useState(0);
28   const [grandTotal, setGrandTotal] = useState(0);
29   const [selectedCustomer, setSelectedCustomer] = useState();
30   const [openQRModal, setOpenQRModal] = useState(false)
31   const [date, setDate] = useState(adToBs(format(Date.now(), "yyyy-MM-dd")));
32   const [scannedPackage, setScannedPackage] = useState([]);
33   const [productList, setProductList] = useState([]);
34   const navigate = useNavigate();
35   const handleRead = (code) => {
36     setVal(JSON.parse(code.data));
37   };
38
39   const [val, setVal] = React.useState();
40
41
42   const [waitingResponse, setWaitingResponse] = useState(false);
43
44   > const handleDate = ({ bsDate, adDate }) => {-
45   };
46   > const handleView = (e) => {-
47   };
48   > const [items, setItems] = useState([-1]);
49
50   > useEffect(() => {-
51     }, [subTotal, discountValue]);
52
53   > useEffect(() => {-
54     }, [vatAmount]);
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
    
```

Figure 103: Screenshot of code developed for adding invoices.

Edit Invoice

Customer Name: H Tigers Footware

Date: 2079-12-19

Location: New Road, Kathmandu

Items

#	Name	Item code	Quantity	Price	Amount
1	Kappa Socks	97	240	370	88800
2	New Products	98	300	555	166500
3	Emoji Socks	96	100	660	66000
4					

Total: 321300

Discount: 0 % = 0

VAT: 41769

Grand Total: 363069

Update **Back**

Figure 104: Screenshot of developed edit invoice page.

```
src > Pages > Invoice > ↗ UpdateInvoice.js > ↗ UpdateInvoice > ↗ loadData > ↗ response
  1 | import { useState } from 'react';
  2 | import { useAuth } from '../../context/AuthContext';
  3 | import { useNavigate } from 'react-router-dom';
  4 | import axios from 'axios';
  5 |
  6 | const UpdateInvoice = () => {
  7 |   const { id } = useParams();
  8 |   const auth = useAuth();
  9 |   const [dbData, setDbData] = useState([]);
 10 |   const [customerView, setCustomerView] = useState(false);
 11 |   const [productView, setProductView] = useState(false);
 12 |   const [renderApp, setRenderApp] = useState(false);
 13 |   const [customerList, setCustomerList] = useState([]);
 14 |   const [subTotal, setSubTotal] = useState(0);
 15 |   const [discount, setDiscount] = useState(0);
 16 |   const [discountValue, setDiscountValue] = useState(0);
 17 |   const [vatAmount, setVatAmount] = useState(0);
 18 |   const [grandTotal, setGrandTotal] = useState(0);
 19 |   const [selectedCustomer, setSelectedCustomer] = useState();
 20 |   const [date, setDate] = useState();
 21 |   const [productList, setProductList] = useState([]);
 22 |   const [show, setShow] = useState(false);
 23 |   const navigate = useNavigate();
 24 |   const handleDate = ({ bsDate, adDate }) => {-
 25 |     const handleView = (e) => {-
 26 |       e.preventDefault();
 27 |       const date = new Date(adDate);
 28 |       const year = date.getFullYear();
 29 |       const month = date.getMonth() + 1;
 30 |       const day = date.getDate();
 31 |       const formattedDate = `${year}-${month}-${day}`;
 32 |       setShow(true);
 33 |       setDate(formattedDate);
 34 |     };
 35 |   };
 36 |   const [items, setItems] = useState([-1]);
 37 |
 38 |   useEffect(() => {-
 39 |     const [subTotal, discountValue] = [
 40 |       items.reduce((acc, item) => acc + item.price * item.quantity, 0),
 41 |       items.filter(item => item.discount).length / items.length
 42 |     ];
 43 |     const [vatAmount] = [
 44 |       subTotal * 0.13
 45 |     ];
 46 |     const [grandTotal] = [
 47 |       subTotal + vatAmount
 48 |     ];
 49 |     const [discount] = [
 50 |       subTotal - (subTotal * discountValue)
 51 |     ];
 52 |     setSubTotal(subTotal);
 53 |     setVatAmount(vatAmount);
 54 |     setSelectedCustomer(items[0].customer);
 55 |     setDate(date);
 56 |     setGrandTotal(grandTotal);
 57 |     setDiscount(discount);
 58 |     setDiscountValue(discountValue);
 59 |     setItems(items);
 60 |   }, [subTotal, discountValue]);
 61 |
 62 |   useEffect(() => {-
 63 |     const [discountValue] = [
 64 |       items.filter(item => item.discount).length / items.length
 65 |     ];
 66 |     setDiscountValue(discountValue);
 67 |   }, [discountValue]);
 68 |
 69 |   useEffect(() => {-
 70 |     const [vatAmount] = [
 71 |       subTotal * 0.13
 72 |     ];
 73 |     setVatAmount(vatAmount);
 74 |   }, [subTotal]);
 75 |
 76 |   const handleChange = (event, index) => {-
 77 |     const value = event.target.value;
 78 |     const item = items[index];
 79 |     const updatedItem = { ...item, quantity: value };
 80 |     const updatedItems = [...items];
 81 |     updatedItems[index] = updatedItem;
 82 |     setItems(updatedItems);
 83 |   };
 84 |
 85 |   const handleFocus = (e) => {-
 86 |     e.preventDefault();
 87 |   };
 88 |   const addFields = (index, value, event) => {-
 89 |     const item = items[index];
 90 |     const updatedItem = { ...item, quantity: value };
 91 |     const updatedItems = [...items];
 92 |     updatedItems[index] = updatedItem;
 93 |     setItems(updatedItems);
 94 |     event.preventDefault();
 95 |   };
 96 |
 97 |   const notifyError = (msg) => toast.error(`${msg}`);
 98 |   const notify = () => toast.success("Invoice added successfully");
 99 |
100 |   const loadData = async () => {
101 |     try {
102 |       const response = await Promise.all([
103 |         axios.get(`${auth?.baseUrl}/api/customer-list`),
104 |         axios.get(`${auth?.baseUrl}/api/product-list`),
105 |       ]);
106 |       const customers = response[0].data;
107 |       const products = response[1].data;
108 |       setCustomerList(customers);
109 |       setProductList(products);
110 |     } catch (error) {
111 |       console.error(error);
112 |     }
113 |   };
114 |
115 |   const [customerView, setCustomerView] = useState(false);
116 |   const [productView, setProductView] = useState(false);
117 |   const [renderApp, setRenderApp] = useState(false);
118 |   const [customerList, setCustomerList] = useState([]);
119 |   const [subTotal, setSubTotal] = useState(0);
120 |   const [discount, setDiscount] = useState(0);
121 |   const [discountValue, setDiscountValue] = useState(0);
122 |   const [vatAmount, setVatAmount] = useState(0);
123 |   const [grandTotal, setGrandTotal] = useState(0);
124 |   const [selectedCustomer, setSelectedCustomer] = useState();
125 |   const [date, setDate] = useState();
126 |   const [productList, setProductList] = useState([]);
127 |   const [show, setShow] = useState(false);
128 |   const [items, setItems] = useState([-1]);
129 |   const [handleChange, setHandleChange] = useState();
130 |   const [handleFocus, setHandleFocus] = useState();
131 |   const [addFields, setAddFields] = useState();
132 | }
```

Figure 105: Screenshot of code developed for editing invoices.

Add Customer

Home / Customers / Add Customer

Customer Name

Phone

PAN Number

District

Location

Email

Opening Balance

Add Back

Figure 106: Screenshot of developed add customer page.

```
src > Pages > Customer > AddCustomer.js > (8) AddCustomer > (8) onSubmit
0 | import { Select } from 'react-select';
1 | const AddCustomer = () => {
2 |   const navigate = useNavigate();
3 |   const notify = () => toast.success("Customer added successfully");
4 |   const auth = useAuth();
5 |   const [districts, setDistricts] = useState([]);
6 |   const [renderApp, setRenderApp] = useState(false);
7 |   const notifyError = (msg) => toast.error(`${msg}`);
8 |
9 |   const {
10 |     register,
11 |     control,
12 |     setError,
13 |     handleSubmit,
14 |     formState: { errors },
15 |   } = useForm();
16 |   const loadData = async () => {
17 |     try {
18 |       const response = await axios.get(`${auth?.baseURL}/api/districts`);
19 |       if (response.status === 200 && setRenderApp(true); setDistricts(response?.data?.data) )
20 |     } catch (err) {
21 |       console.log(err);
22 |     }
23 |   };
24 |   useEffect(() => {
25 |     loadData();
26 |   }, [ ]);
27 |
28 |   const onSubmit = async (data) => {
29 |     try {
30 |       const response = await axios.post(`${auth?.baseURL}/api/customers`, data);
31 |       if (response.status === 201) {
32 |         navigate('/customers');
33 |         notify();
34 |       }
35 |     } catch (err) {
36 |       setError(err);
37 |     }
38 |   };
39 |
40 |   return (
41 |     <div>
42 |       {renderApp &&
43 |        <div className='px-7 font-secondary py-10'>
44 |          <ToastContainer className='text-sm text-grey' />
45 |          <div className='font-bold text-2xl'>Add Customer</div>
46 |          <div className='breadcum-container flex items-center mt-1 text-xs font-thin text-grey'>...
47 |          </div>
48 |          <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
49 |            <form onSubmit={handleSubmit(onSubmit)} action='>
50 |              <div className='flex flex-col mb-5'>...
51 |              </div>
52 |              <div className='grid grid-cols-2'>...
53 |              </div>
54 |              <div className='flex flex-col mb-5'>
55 |                <label className='text-sm' htmlFor='district'>
56 |                  District
57 |                </label>
58 |                <Controller
59 |                  name='district'
60 |                  control={control}
61 |                  rules={{ required: true }}
62 |                  render={({ field }) => (
63 |                    <Select
64 |                      {...field}
65 |                      id='district-select'
66 |                      options={districts} className='mt-3 text-xs rounded-lg' />
67 |                  )} />
68 |              </div>
69 |            </form>
70 |          </div>
71 |        </div>
72 |      </div>
73 |    );
74 |  }
75 |
76 |  </div>
77 |
```

Figure 107: Screenshot of code developed for adding customer.

ID	Name	Location	Phone	District	Balance	Status	Action
43	H Tigers Footware	New Road, Kathmandu	98189878432	Baitadi	-690242.90	● Active	⋮
44	AB Store	Birjung, Nepal	98761222122	Chitawan	-24521.00	● Active	⋮
41	B.N Traders	New Road, Kathmandu	015329953	Bhaktapur	-40873.55	● Not Active	⋮
40	Singh General Store	Dhangadhi	98232133133	Kailali	-665050.20	● Active	⋮
42	Lumbini Fancy Store	Lumbini, Nepal	9821612782	Syangja	-107937.60	● Active	⋮

Page 1 of 1 | Go to page: Total: 5

+ Add Customer | Search... | ⋮

Figure 108: Screenshot of developed customer page.

```

8  const Customer = () => {
9    const location = useLocation();
10   const auth = useAuth();
11   const [allCustomers, setAllCustomers] = useState([]);
12   const [renderApp, setRenderApp] = useState(false);
13   const loadData = async () => {
14     try {
15       const response = await axios.get(
16         `${auth?.baseUrl}/api/customers`
17       );
18       response?.status === 200 && setAllCustomers(response?.data?.data);
19       setRenderApp(true);
20     } catch (err) {
21       console.log(err);
22     }
23   };
24   useEffect(() => [
25     loadData(),
26     console.log("rerender"),
27   ], [location]);
28
29 >   const notify = () => { ... };
30
31   const notifyError = (msg) => toast.error(`#${msg}`);
32   const handleDelete = async (data) => { ... };
33
34 >
35   return (
36     <div className='px-7 font-secondary py-10'>
37       <ToastContainer className='text-sm text-grey' />
38       <div className='flex justify-between'>
39         <div className='flex items-center'>
40           <Link
41             to='/customer/add-customer'
42             className='bg-primary text-white p-2 py-3 mr-3 text-sm flex items-center font-main rounded-lg'>
43             <span className='material-symbols-outlined text-sm mr-2'>add</span>
44             <span>Add Customer</span>
45           </Link>
46         </div>
47         <div className='product-table'>
48           {renderApp && (
49             <Table handleDelete={handleDelete} data={allCustomers} loadData={loadData} />
50           )}
51         </div>
52       </div>
53     </div>
54   );
55
56   export default Customer;

```

Figure 109: Screenshot of code developed for viewing customer list.

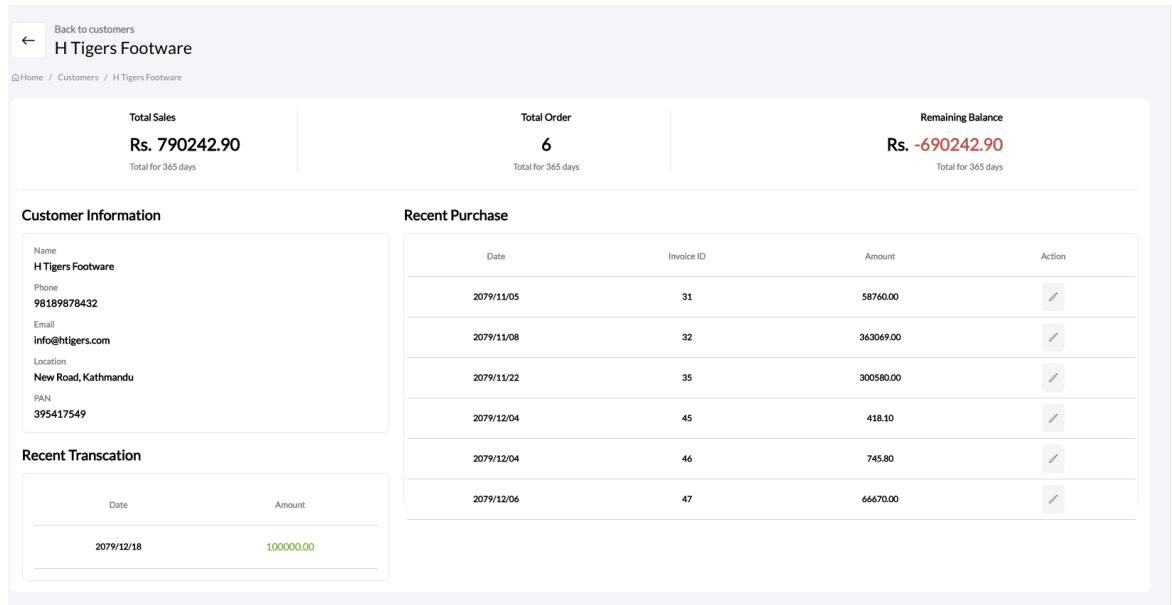


Figure 110: Screenshot of developed customer information page.

```

src > Pages > Customer > js CustomerInfo.js > CustomerInfo > loadData
  9 | import { useState, useEffect } from 'react';
10 | import { useAuth } from '../../../../../Components/Authentication/auth';
11 | const CustomerInfo = () => {
12 |   const { id } = useParams();
13 |   const auth = useAuth();
14 |   const [customerData, setCustomerData] = useState([]);
15 |   const [transaction, setTransaction] = useState([]);
16 |   const [salesReport, setSalesReport] = useState([]);
17 |   const [money, setMoney] = useState([]);
18 |   const [renderApp, setRenderApp] = useState(false);
19 |   const loadData = async () => {
20 |     try {
21 |       const response = await Promise.all([
22 |         axios.get(`${auth?.baseUrl}/api/customer-transaction/${id}`),
23 |         axios.get(`${auth?.baseUrl}/api/customer/${id}`),
24 |         axios.get(`${auth?.baseUrl}/api/customer-sales-report/${id}`),
25 |         axios.get(`${auth?.baseUrl}/api/money-transaction/${id}`)
26 |       ]);
27 |       setTransaction(response[0].data.data);
28 |       setCustomerData(response[1].data.data);
29 |       setSalesReport(response[2].data.data[0]);
30 |       setMoney(response[3].data.data);
31 |       setRenderApp(true);
32 |     } catch (err) {
33 |       console.log(err);
34 |     };
35 |   };
36 |   useEffect(() => {
37 |     loadData();
38 |   }, []);
39 |   return (
40 |     <div>
41 |       {renderApp && (
42 |         <>
43 |           <div className='px-7 font-secondary py-10'>
44 |             <div className='flex justify-between'>
45 |               <div className='flex'>--</div>
46 |               <div>--</div>
47 |             </div>
48 |             <div className='breadcrumb-container flex items-center mt-4 text-xs font-thin text-grey'>--</div>
49 |           </div>
50 |           <div className='bg-white rounded-lg p-4 mt-5'>--</div>
51 |         </div>
52 |       );
53 |     );
54 |   };
55 |   export default CustomerInfo;
56 |

```

Figure 111: Screenshot of code developed for viewing customer information.

Edit Customer

Home / Customers / Edit Customer

Customer Name
H Tigers Footware

Phone
98189878432

PAN Number
395417549

Location
New Road, Kathmandu

Email
info@htigers.com

Save **Back**

Figure 112: Screenshot of developed edit customer page.

```

src > Pages > Customer > js EditCustomer.js > [e] EditCustomer > [e] onSubmit
  1 | import { useState } from '...';
  2 | import { useAuth } from '../../../../../Components/Authentication/auth';
  3 |
  4 | const EditCustomer = () => {
  5 |   const { id } = useParams();
  6 |   const navigate = useNavigate();
  7 |   const [oldData, setOldData] = useState([]);
  8 |   const [renderApp, setRenderApp] = useState(false);
  9 |
 10 |   const auth = useAuth();
 11 |
 12 |   const notify = () => toast.success("Customer updated successfully");
 13 |   const notifyError = (msg) => toast.error(`${msg}`);
 14 |
 15 |   const [form, setForm] = useForm();
 16 |
 17 |   const [onSubmit] = useState(async (data) => [
 18 |     ...
 19 |   ]);
 20 |
 21 |   const loadData = async () => {
 22 |     ...
 23 |   };
 24 |
 25 |   useEffect(() => {
 26 |     loadData();
 27 |   }, []);
 28 |
 29 |   return (
 30 |     <div>
 31 |       {renderApp && (
 32 |         <div className='px-7 font-secondary py-10'>
 33 |           <ToastContainer className='text-sm text-grey' />
 34 |           <div className='font-bold text-2xl'>Edit Customer</div>
 35 |           <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>...
 36 |         </div>
 37 |         {oldData && (
 38 |           <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
 39 |             <form onSubmit={handleSubmit(onSubmit)} action='>
 40 |               <div className='flex flex-col mb-5'>...
 41 |                 </div>
 42 |                 <div className='grid grid-cols-2'>...
 43 |                   <div className='flex flex-col mb-5'>...
 44 |                     </div>
 45 |                     <div className='grid grid-cols-2'>...
 46 |                       <div className='container text-end flex items-center'>...
 47 |                         </div>
 48 |                       </form>
 49 |                     </div>
 50 |                   </div>
 51 |                 </div>
 52 |               </div>
 53 |             </div>
 54 |           </div>
 55 |         );
 56 |       );
 57 |     );
 58 |   );
 59 |
 60 |   export default EditCustomer;

```

Figure 113: Screenshot of code developed for editing customers.

The screenshot shows a web application interface for managing transactions. At the top right is a purple button labeled '+ Add Transaction'. Below it is a search bar with placeholder text 'Search...'. A navigation bar at the bottom includes buttons for 'Previous' and 'Next', and arrows for navigating through pages.

Date	Transaction ID	Party Name	Amount	Action
2079/12/18	122	H Tigers Footware	100000.00	⋮

Page 1 of 1 | Go to page: Total: 1

Figure 114: Screenshot of developed transaction page.

```

src > Pages > Transaction > js Transaction.js > Transaction > loadData > response
    import { useAuth } from '../../../../../components/authentication/auth';
    const Transaction = () => {
      const location = useLocation();
      const auth = useAuth();
      const [allTransaction, setAllTransaction] = useState([]);
      const [renderApp, setRenderApp] = useState(false);
      const loadData = async () => {
        try {
          const response = await axios.get(`${auth?.baseURL}/api/get-transaction`);
          response?.status === 200 && setAllTransaction(response?.data?.data);
          setRenderApp(true);
        } catch (err) {
          console.log(err);
        }
      };
      useEffect(() => {
        loadData();
      }, [location]);
    }

    return (
      <div className='px-7 font-secondary py-10'>
        <ToastContainer className='text-sm text-grey' />
        <div className='flex justify-between'>
          <div className='>
            <div className='font-bold text-2xl'>Transaction</div>
            <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
              <div className='first flex items-center'>
                <span className='material-symbols-outlined text-sm'>home</span>
                <span className='>Home</span>
              </div>
              <div className='divider mx-2' />
              <div className='second'>Transaction</div>
            </div>
          </div>
          <div className='>
            <Link
              to='/transaction/add-transaction'
              className='bg-primary text-white p-2 text-sm flex items-center font-main rounded-lg'>
              <span className='material-symbols-outlined text-sm mr-2'>add</span>
              <span>Add Transaction</span>
            </Link>
          </div>
        </div>
        <div className='Transaction-table'>
          {renderApp && <Table data={allTransaction} loadData={loadData} />}
        </div>
      </div>
    );
  };

  export default Transaction;

```

Figure 115: Screenshot of code developed for viewing transaction list.

Add Transaction

Home > Transactions > Add Transaction

Customer Name	Date: 2079-12-19
Type Customer here	
Amount	
Type Amount here	
Description	
Type Description here	

Add **Back**

Figure 116: Screenshot of developed add transaction page.

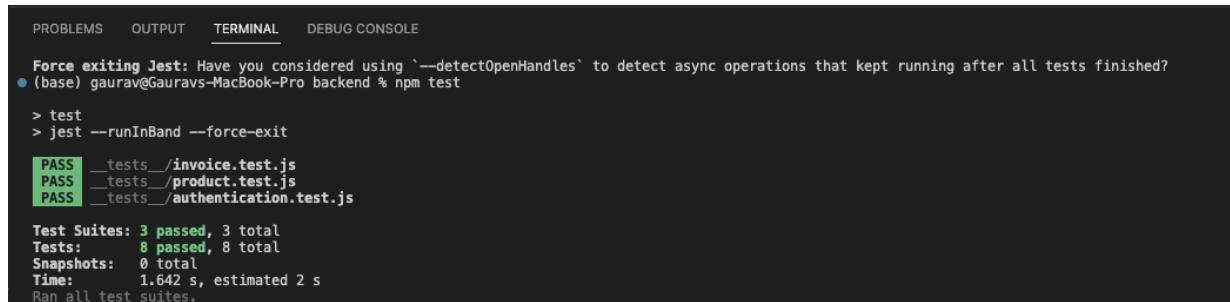
```

src > Pages > Transcation > AddTransaction.js > [o] AddTransaction
13 < const AddTransaction = () => [
14   const [allCustomer, setAllCustomer] = useState([]);
15   const [renderApp, setRenderApp] = useState(false);
16   const [selectedCustomer, setSelectedCustomer] = useState();
17   const [date, setDate] = useState(adToBs(format(Date.now(), "yyyy-MM-dd")));
18   const auth = useAuth()
19   const navigate = useNavigate();
20   const handleDate = ({ bsDate, adDate }) => { ...
21 };
22   const notify = () => toast.success("Transaction added successfully");
23   const {
24     ...,
25     ...,
26     ...,
27     ...,
28     ...,
29     ...,
29   } = useForm();
30   const loadData = async () => {
31     try {
32       const response = await axios.get(
33         `${auth?.baseUrl}/api/customer-list`
34       );
35       setAllCustomer(response?.data?.data);
36       setRenderApp(true);
37     } catch (err) {
38       console.log(err);
39     }
40   };
41   useEffect(() => {
42     loadData();
43   }, []);
44   const onSubmit = async (data) => { ...
45 };
46   return (
47     <div>
48       {renderApp && (
49         ...
50         <div className='px-7 font-secondary py-10'>
51           <ToastContainer className='text-sm text-grey' />
52           <div className='font-bold text-2xl'>Add Transaction</div>
53           <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>...
54           </div>
55           <div className='mt-5'>
56             <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
57               <form onSubmit={handleSubmit(onSubmit)} action='...'>
58                 <div className='flex items-center'>...
59                 <div className='mt-5'>...
60                 </div>
61                 <div className='container text-end flex items-center'>...
62                 </div>
63               </form>
64             </div>
65           </div>
66           </div>
67         </div>
68       )} ...
69     </div>
70   );
71 
```

Figure 117: Screenshot of code developed for adding transaction.

3.7.3.3. Testing

a) API Testing



```

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?
● (base) gaurav@Gauravs-MacBook-Pro backend % npm test

> test
> jest --runInBand --force-exit

PASS  __tests__/invoice.test.js
PASS  __tests__/product.test.js
PASS  __tests__/authentication.test.js

Test Suites: 3 passed, 3 total
Tests:     8 passed, 8 total
Snapshots: 0 total
Time:      1.642 s, estimated 2 s
Ran all test suites.

```

Figure 118: Screenshot of API testing result

b) Cypress Testing

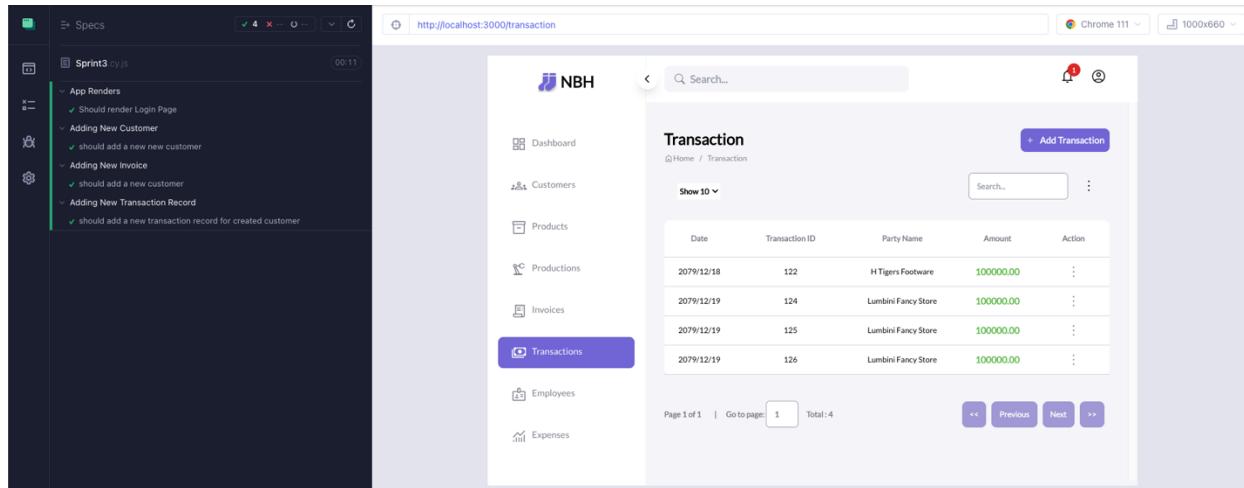


Figure 119: Screenshot of frontend testing result

3.7.3.4. Sprint Retrospective

The sprint goal was achieved, but there were challenges with planning and testing. The individual plans to improve planning by prioritizing tasks and testing by conducting more in-depth testing. I will also maintain open communication with stakeholders.

3.7.4. Sprint 4 – Employee Management System

3.7.4.1. Sprint Planning

This project aims to develop an Employee Management System that stores personal information, tracks attendance, and manages employee status. The tasks include custom employee views, Figma mockups, and database table creation, with deliverables including a tested attendance feature and Figma mockups for adding employees and attendance. Mitigation strategies, such as breaking down tasks, creating a detailed plan, conducting stand-ups, have been put in place to address risks of complexity and quality standards.

3.7.4.2. Development

a) Figma Mockup

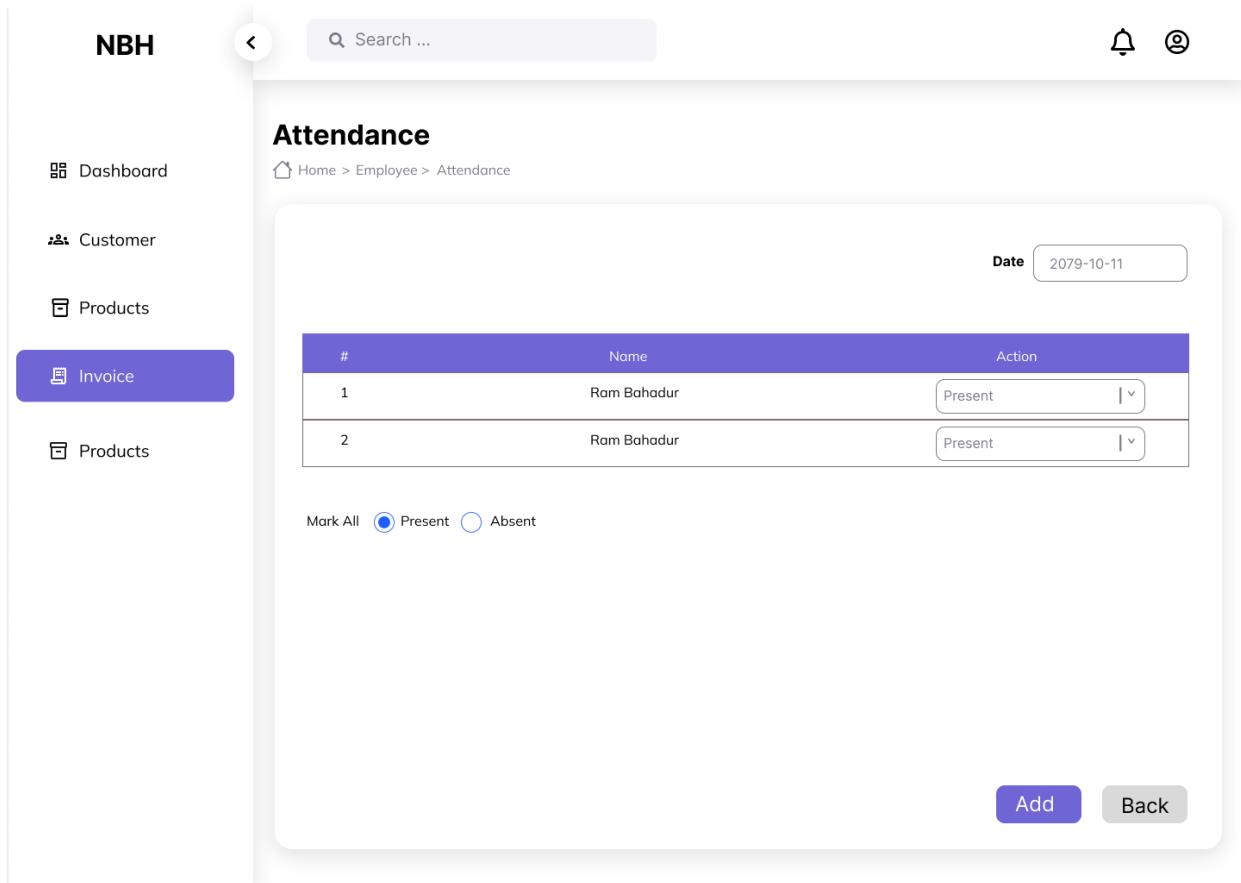


Figure 120: Figma mockup for adding attendance page.

The image shows a Figma mockup of an 'Add Employee' form. At the top left is a sidebar with the logo 'NBH' and five menu items: Dashboard, Customer, Products, Invoice, and Employee, with 'Employee' highlighted in purple. A search bar at the top right contains the placeholder 'Search ...'. The main content area has a title 'Add Employee' and a breadcrumb navigation 'Home > Employee > Add Employee'. The form fields are grouped into sections: 'Employee Name' (a single input field), 'Phone' (input field) and 'PAN Number' (input field), 'Location' (input field), 'Salary' (input field), and 'Joined Date' (input field). At the bottom right are two buttons: a purple 'Add' button and a grey 'Back' button.

Figure 121: Figma mockup for adding employee.

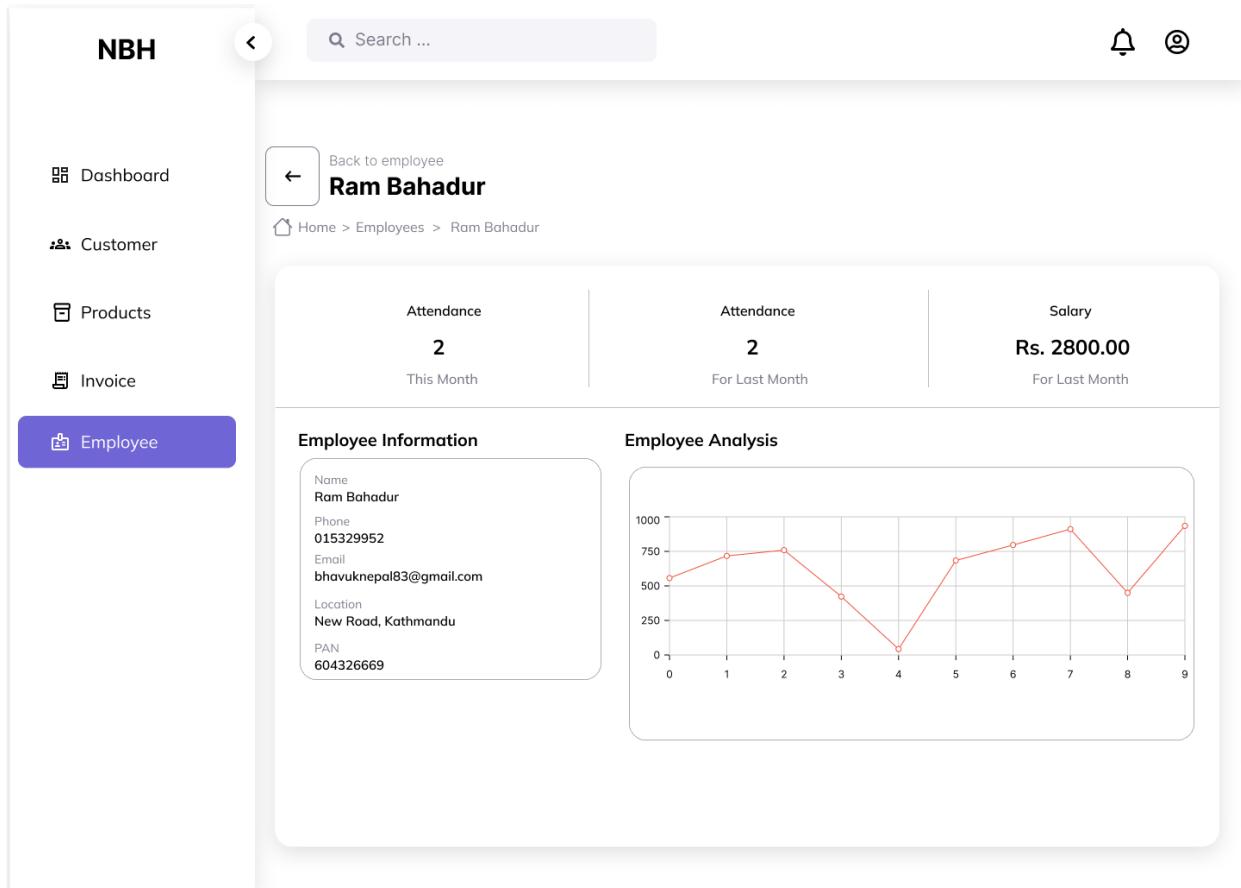


Figure 122: Figma mockup for viewing employee information.

The image shows a Figma mockup of a web application interface for managing employees. On the left is a sidebar with icons for Dashboard, Customer, Products, Invoice, and Employee (which is highlighted with a purple background). The main area has a header with a search bar, a bell icon, and a user icon. It displays a breadcrumb path: Home > Employee. Below this is a table with columns: ID, Name, Location, Phone, Status, and Action. Two rows of data are shown, both with ID 1, Name 001, Location B.N Traders, Phone 22222, and Status Active. A vertical ellipsis in the Action column of the second row opens a dropdown menu with three options: View Employee, Edit Employee, and Archive.

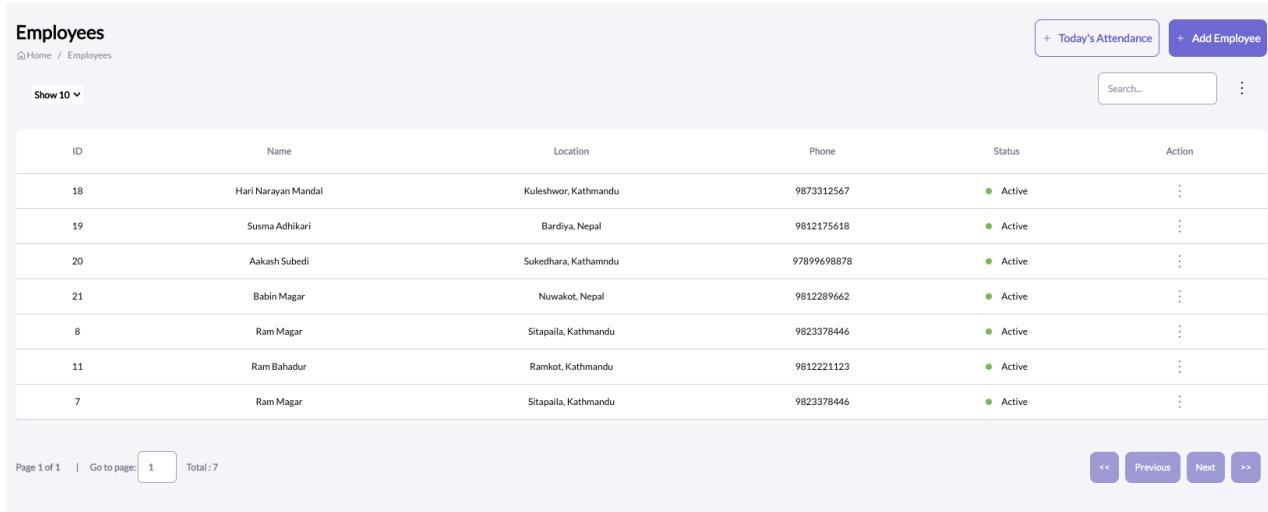
ID	Name	Location	Phone	Status	Action
1	001	B.N Traders	22222	Active	⋮
1	001	B.N Traders	22222	Active	⋮

Page 1 of 1 | Go to page:

<< Previous Next >>

Figure 123: Figma mockup for viewing employee list page.

b) Frontend development

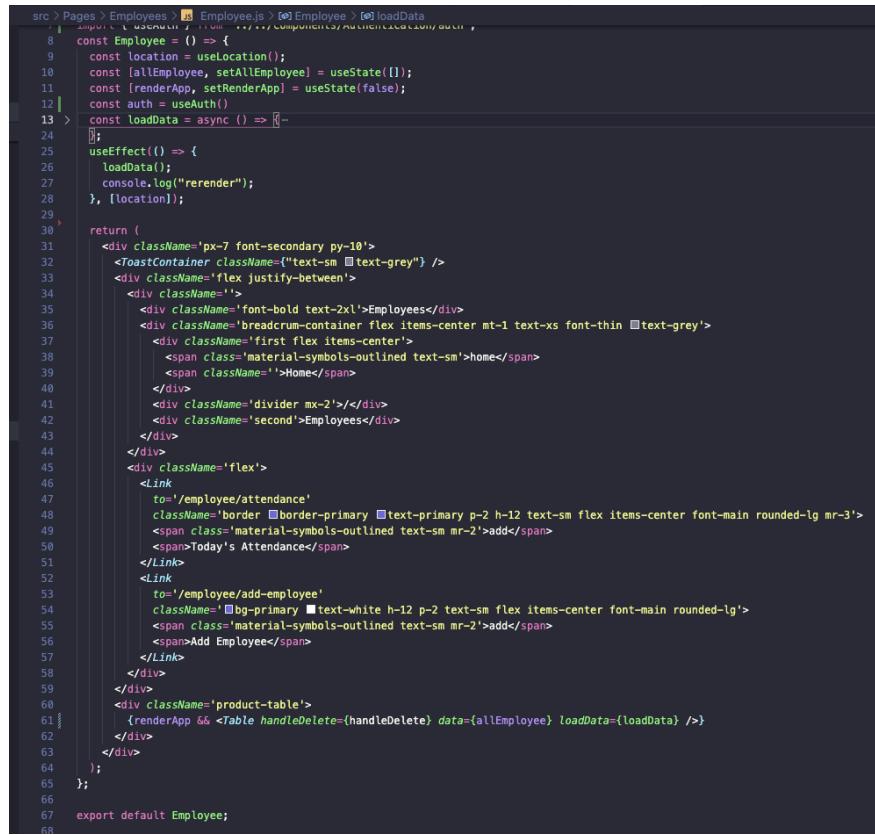


The screenshot shows a web application interface for managing employees. At the top right are buttons for "Today's Attendance" and "Add Employee". Below them is a search bar labeled "Search...". The main content is a table with the following data:

ID	Name	Location	Phone	Status	Action
18	Hari Narayan Mandal	Kuleshwor, Kathmandu	9873312567	Active	⋮
19	Susma Adhikari	Bardiya, Nepal	9812175618	Active	⋮
20	Aakash Subedi	Sukedhara, Kathmandu	97899698878	Active	⋮
21	Babin Magar	Nuwakot, Nepal	9812289662	Active	⋮
8	Ram Magar	Sitapalika, Kathmandu	9823378446	Active	⋮
11	Ram Bahadur	Ramkot, Kathmandu	9812221123	Active	⋮
7	Ram Magar	Sitapalika, Kathmandu	9823378446	Active	⋮

At the bottom left, it says "Page 1 of 1 | Go to page: 1 Total: 7". At the bottom right are navigation buttons: <<, Previous, Next, >>.

Figure 124: Screenshot of developed employee list page



```

src > Pages > Employees > Employee.js > Employee > loadData
1 import { useState, useEffect } from 'react';
2 import { useAuth, useLocation } from '../../Components/Authentications/auth';
3 const Employee = () => {
4   const location = useLocation();
5   const [allEmployee, setAllEmployee] = useState([]);
6   const [renderApp, setRenderApp] = useState(false);
7   const auth = useAuth();
8   const loadData = async () => [
9     ...
10   ];
11   useEffect(() => {
12     loadData();
13     console.log("rerender");
14   }, [location]);
15
16   return (
17     <div className='px-7 font-secondary py-10'>
18       <ToastContainer className='text-sm text-grey' />
19       <div className='flex justify-between'>
20         <div className='flex'>
21           <div className='font-bold text-2xl'>Employees</div>
22           <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
23             <div className='first flex items-center'>
24               <span className='material-symbols-outlined text-sm'>home</span>
25               <span>Home</span>
26             </div>
27             <div className='divider mx-2'></div>
28             <div className='second'>Employees</div>
29           </div>
30         </div>
31         <div className='flex'>
32           <Link
33             to='/employee/attendance'
34             className='border border-primary text-primary p-2 h-12 text-sm flex items-center font-main rounded-lg mr-3'>
35               <span className='material-symbols-outlined text-sm mr-2'>add</span>
36               <span>Today's Attendance</span>
37             </Link>
38           <Link
39             to='/employee/add-employee'
40             className='bg-primary text-white h-12 p-2 text-sm flex items-center font-main rounded-lg'>
41               <span className='material-symbols-outlined text-sm mr-2'>add</span>
42               <span>Add Employee</span>
43             </Link>
44           </div>
45         </div>
46       </div>
47       <div className='product-table'>
48         <Table handleDelete={handleDelete} data={allEmployee} loadData={loadData} />
49       </div>
50     </div>
51   );
52 }
53 export default Employee;
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```

Figure 125: Screenshot of developed code for viewing employee list.

Add Employee

[Home](#) / [Employees](#) / Add Employee

Employee Name		
<input type="text" value="Type name here"/>		
Phone	PAN Number	
<input type="text" value="Type phone number here"/>	<input type="text" value="Type PAN here"/>	
Location		
<input type="text" value="Type address here"/>		
Salary	Joined Date	
<input type="text" value="Type salary here"/>	<input type="text" value="2079-12-21"/>	

Figure 126: Screenshot of developed code for adding new employee.

```

src > Pages > Employees > AddEmployee.js > AddEmployee
13 const AddEmployee = () => {
14   const navigate = useNavigate();
15   const notify = () => toast.success("Employee added successfully");
16   const notifyError = (msg) => toast.error(`${msg}`);
17   const [date, setDate] = useState(adDate(format(Date.now(), "yyyy-MM-dd")));
18   const auth = useAuth();
19   const handleDate = ({ bsDate, adDate }) => {-
20     };
21   const {-
22   } = useForm();
23   const onSubmit = async (data) => {-
24     };
25   return [(
26     <div>
27       <ToastContainer className="text-sm text-grey" />
28       <div className="breadrum-container flex items-center mt-1 text-xs font-thin text-grey">
29         <div className="first flex items-center">
30           <span className="material-symbols-outlined text-sm">home</span>
31           <span>Home</span>
32         </div>
33         <div className="divider mx-2"></div>
34         <div className="second">Employees</div>
35         <div className="divider mx-2"></div>
36         <div className="second">Add Employee</div>
37       </div>
38       <div className="bg-white mt-5 p-8 rounded-md border font-thin">
39         <form onSubmitted={handleSubmit(onSubmit)} action="">
40           <div className="flex flex-col mb-5">-
41           <div className="grid grid-cols-2">-
42             <div>-
43             <div className="flex flex-col mb-5">-
44               <div className="grid grid-cols-2">-
45                 <div className="container text-end flex items-center">
46                   <button className="bg-primary ml-auto text-white py-2 px-5 text-md font-bold flex items-center font-main rounded-lg">
47                     Add
48                   </button>
49                   <a href="/employee" className="ml-2 bg-gray-200 py-2 px-5 text-md font-bold flex items-center font-main rounded-lg" to="/employee">
50                     Back
51                   </a>
52                 </div>
53               </div>
54             </div>
55           </div>
56         </div>
57       </form>
58     </div>
59   </div>
60   );
61 }
62 export default AddEmployee;

```

Figure 127: Screenshot of developed code for adding new employees.

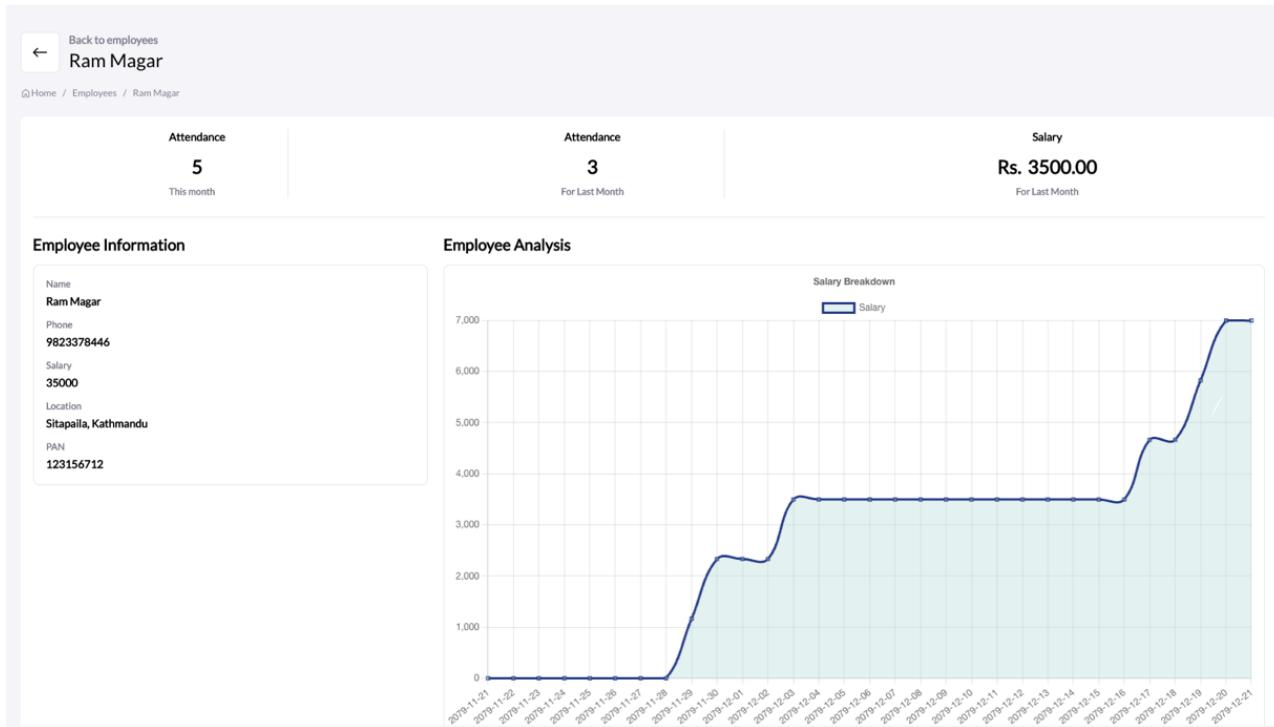


Figure 128: Screenshot of developed employee profile page

```

src > Pages > Employees > ↗ EmployeeInfo.js > ↗ EmployeeInfo > ⚡ useEffect() callback
32  );
33  ✓ const EmployeeInfo = () => {
34    const auth = useAuth();
35    const currentDate = adTobS(format(Date.now(), "yyyy-MM-dd"));
36    const { id } = useParams();
37    const [employeeInfo, setEmployeeInfo] = useState([]);
38    const [currentAttendance, setCurrentAttendance] = useState(0);
39    const [lastAttendance, setLastAttendance] = useState(0);
40    const endDate = adTobS(format(Date.now(), "yyyy-MM-dd"));
41    const startDate = adTobS(format(subDays(Date.now(), 30), "yyyy-MM-dd"));
42    const [employeeAttendance, setEmployeeAttendance] = useState();
43    const [renderApp, setRenderApp] = useState(false);
44    const [data, setData] = useState();
45    const options = {};
46  };
47
48  ✓ useEffect(() => [
49    id,
50  ]);
51
52
53  ✓ useEffect(() => {
54    const [employeeAttendance] = useState();
55    return (
56      <div>
57        {renderApp &&
58          <div className='px-7 font-secondary py-10'>
59            <div className='flex justify-between'>
60              <div className='breadcrumb-container flex items-center mt-4 text-xs font-thin text-gray'>
61                <a href="#">Home</a>
62                <a href="#">Employees</a>
63                <a href="#">Employee Info</a>
64              </div>
65              <div className='bg-white rounded-lg p-4 mt-5'>
66                <div className='small-customer-summary flex justify-around border-b pb-6'>
67                  <div>
68                    <div>
69                      <div>
70                        <div>
71                          <div>
72                            <div>
73                              <div>
74                                <div>
75                                  <div>
76                                    <div>
77                                      <div>
78                                        <div>
79                                          <div>
80                                            <div>
81                                              <div>
82                                                <div>
83                                                  <div>
84                                                    <div>
85                                                      <div>
86                                                        <div>
87                                                          <div>
88                                                            <div>
89                                                              <div>
90                                                                <div>
91                                                                  <div>
92                                                                    <div>
93                                                                      <div>
94                                                                        <div>
95                                                                          <div>
96                                                                            <div>
97                                                                              <div>
98                                                                                <div>
99                                                                                  <div>
100                                                                 <div>
101                                                                 <div>
102                                                                 <div>
103                                                                 <div>
104                                                                 <div>
105                                                                 <div>
106                                                                 <div>
107                                                                 <div>
108                                                                 <div>
109                                                                 <div>
110                                                                 <div>
111                                                                 <div>
112                                                                 <div>
113                                                                 <div>
114                                                                 <div>
115                                                                 <div>
116                                                                 <div>
117                                                                 <div>
118                                                                 <div>
119                                                                 <div>
120                                                                 <div>
121                                                                 <div>
122                                                                 <div>
123                                                                 <div>
124                                                                 <div>
125                                                                 <div>
126                                                                 <div>
127                                                                 <div>
128                                                                 <div>
129                                                                 <div>
130                                                                 <div>
131                                                                 <div>
132                                                                 <div>
133                                                                 <div>
134                                                                 <div>
135                                                                 <div>
136                                                                 <div>
137                                                                 <div>
138                                                                 <div>
139                                                                 <div>
140                                                                 <div>
141                                                                 <div>
142                                                                 <div>
143                                                                 <div>
144                                                                 <div>
145                                                                 <div>
146                                                                 <div>
147                                                                 <div>
148                                                                 <div>
149                                                                 <div>
150                                                                 <div>
151                                                                 <div>
152                                                                 <div>
153                                                                 <div>
154                                                                 <div>
155                                                                 <div>
156                                                                 <div>
157                                                                 <div>
158                                                                 <div>
159                                                                 <div>
160                                                                 <div>
161                                                                 <div>
162                                                                 <div>
163                                                                 <div>
164                                                                 <div>
165                                                                 <div>
166                                                                 <div>
167                                                                 <div>
168                                                                 <div>
169                                                                 <div>
170                                                                 <div>
171                                                                 <div>
172                                                                 <div>
173                                                                 <div>
174                                                                 <div>
175                                                                 <div>
176                                                                 <div>
177                                                                 <div>
178                                                                 <div>
179                                                                 <div>
180                                                                 <div>
181                                                                 <div>
182                                                                 <div>
183                                                                 <div>
184                                                                 <div>
185                                                                 <div>
186                                                                 <div>
187                                                                 <div>
188                                                                 <div>
189                                                                 <div>
190                                                                 <div>
191                                                                 <div>
192                                                                 <div>
193                                                                 <div>
194                                                                 <div>
195                                                                 <div>
196                                                                 <div>
197                                                                 <div>
198                                                                 <div>
199                                                                 <div>
200                                                                 <div>
201                                                                 <div>
202                                                                 <div>
203                                                                 <div>
204                                                                 <div>
205                                                                 <div>
206                                                                 <div>
207                                                                 <div>
208                                                                 <div>
209                                                                 <div>
210                                                                 <div>
211                                                                 <div>
212                                                                 <div>
213                                                                 <div>
214                                                                 <div>
215                                                                 <div>
216                                                                 <div>
217                                                                 <div>
218                                                                 <div>
219                                                                 <div>
220                                                                 <div>
221                                                                 <div>
222                                                                 <div>
223                                                                 <div>
224                                                                 <div>
225                                                                 <div>
226                                                                 <div>
227                                                                 <div>
228                                                                 <div>
229                                                                 <div>
230   export default EmployeeInfo;

```

Figure 129: Screenshot of developed code for employee profile page.

3.7.4.3. Testing

a) API Testing

```

PROBLEMS      OUTPUT      TERMINAL      DEBUG CONSOLE

PASS __tests__\employee.test.js
CRUD employee
✓ Getting All Employee (14 ms)
✓ Getting Employee Analysis (8 ms)
✓ Editing a Employee (128 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.95 s, estimated 1 s
Ran all test suites matching /__tests__\employee.test.
Force exiting Jest: Have you considered using '--detect

```

Figure 130: Screenshot of test case result developed for Sprint 4

b) Cypress testing

The screenshot shows the Cypress Test Runner interface on the left and a browser window on the right.

Cypress Test Runner:

- Specs: Sprint4 cypress
- App Renders: Should render Login Page, Adding New Employee (should add a new employee, should add a archive the employee), Adding New Attendance Record for employee (should employee attendance for the date)

Browser View:

- Address bar: http://localhost:3000/employee
- Page Title: NBH
- Left sidebar menu: Dashboard, Customers, Products, Productions, Invoices, Transactions, Employees (highlighted in purple), Reports, Expenses.
- Main Content Area:
 - Employees section: + Today's Attendance, + Add Employee
 - Table: Employees (listing 11 rows of employee data)

Figure 131: Screenshot of success result from cypress test case

3.7.4.4. Sprint Retrospective

The sprint goal was to develop an employee management system that records staff attendance and stores personnel information. The system was successfully implemented, and a scheduled task was added for adding employee attendance on Saturdays. However, frontend testing was not performed during this sprint, but it will be conducted in the next sprint. To improve future sprints, the individual will prioritize the most important and high-value items and collect user input often to identify problems, potential solutions, and new feature requests.

3.7.5. Sprint 5 – Role Management System

3.7.5.1. Sprint Planning

The aim of sprint is to build a system based on roles where users with authority can design unique jobs. Making Figma mockups, building the application with a role-based structure, and protecting the API from unauthorized users are all tasks for this sprint. A fully functional application with the ability to manage different roles and restrict user access is one of the deliverables for this sprint. Another is a 404 page for unauthorized users.

3.7.5.2. Development

- Initial Development as Planned

Initially the planned development was to create a role which could be assigned to any individual user of the system. With no pre-defined role in the system, all the roles were meant to be created as per requirement.

The screenshot shows a Figma wireframe for an 'Add Role' interface. On the left, there's a sidebar with 'Company Logo' (empty placeholder), 'Dashboard', 'User', and 'Role'. The main area has a 'Search' bar, a notification bell icon, and a user profile icon. The central part is titled 'Add Role' with a 'Role Name' input field (empty). Below it is a section for 'Available Roles' with checkboxes for 'Products', 'Employees', 'Customers', and 'Money Transaction'. At the bottom right are 'Add' and 'Cancel' buttons.

Figure 132: Initially created role-based system.

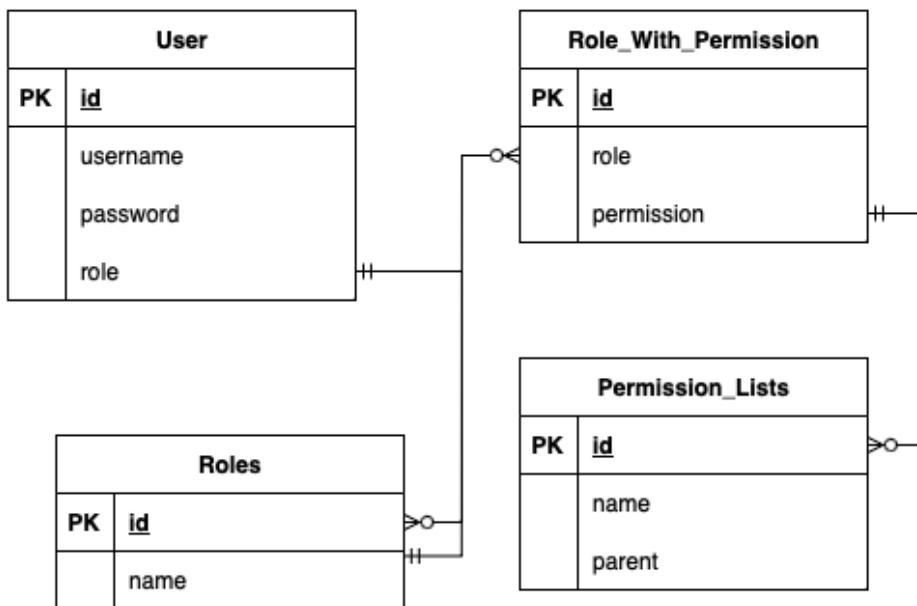


Figure 133: Initially planned ERD for role management system.

- Reason For Change

There was a logical error in the planned process for a project that involved developing an employee management system. The client industry did not want a custom role implementation and found the development process difficult. The supervisor was consulted and suggested creating three pre-defined roles: Admin, Supervisor, and Staff. The permission for the Staff role could be customized as defined in the project proposal.

- Development after change
- a) Figma Mockup

The image shows a Figma mockup of a user creation interface. At the top left is a sidebar with 'NBH' and navigation links for 'Dashboard' and 'Users'. The main area has a header 'Add User' with a breadcrumb 'Home > User > Add User'. It contains fields for 'Username' (placeholder 'Type name here ...'), 'Email' (placeholder 'Type email here ...'), 'Password' (placeholder 'Type Password here ...'), and 'Confirm Password' (placeholder 'Retype Password here ...'). A 'User Type' dropdown menu is shown with 'Type type Here'. Under 'Permissions', three checkboxes are checked: 'Customer', 'Products', and 'Production'. At the bottom right are 'Add' and 'Back' buttons.

Figure 134: Figma mockup for creating user

b) Table design

TABLE NAME			
<code>role_with_permission</code>			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
<code>id</code>	integer	<code>identity</code> ↘	<code>PRIMARY KEY</code> +
Indexes			
INDEX NAME	TYPE	COLUMNS	
<code>untitled_table_pkey</code>	Primary Key Index	<code>id</code>	+ ↗

Figure 135: Table design for Sprint 5 - role_with_permission

TABLE NAME			
<code>users</code>			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
<code>id</code>	integer	<code>sequence</code> ↘ <code>users_id_seq</code>	<code>PRIMARY KEY</code> + ↗
<code>username</code>	text	<code>no default</code> ↘	<code>UNIQUE</code> <code>NOT NULL</code> + ↗
<code>password</code>	text	<code>no default</code> ↘	<code>NOT NULL</code> + ↗
<code>user_type</code>	text	<code>no default</code> ↘	+ ↗
<code>email</code>	text	<code>no default</code> ↘	<code>UNIQUE</code> + ↗
<code>using_default_password</code>	integer	<code>expression</code> ↘ 1	+ ↗
Indexes			
INDEX NAME	TYPE	COLUMNS	
<code>users_pkey</code>	Primary Key Index	<code>id</code>	+ ↗
<code>users_username_key</code>	Unique Index	<code>username</code>	+ ↗
<code>users_email_key</code>	Unique Index	<code>email</code>	+ ↗

Figure 136: Table design for Sprint 5 - users

TABLE NAME			
permission_lists			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity	PRIMARY KEY +
name	text	no default	UNIQUE NOT NULL +
Indexes			
INDEX NAME	TYPE	COLUMNS	
permission_lists_pkey	Primary Key Index	id	
permission_lists_name_key	Unique Index	name	

Figure 137: Table design for Sprint 5 - permission_lists

c) Frontend Development

Add User

Home / Users / Add User

User Name

testdsa

Email

Type email here

Password

Confirm Password

Retype password here

User Type

Staff

Available Roles

Products Customers Transactions

Invoices Productions

Add Back

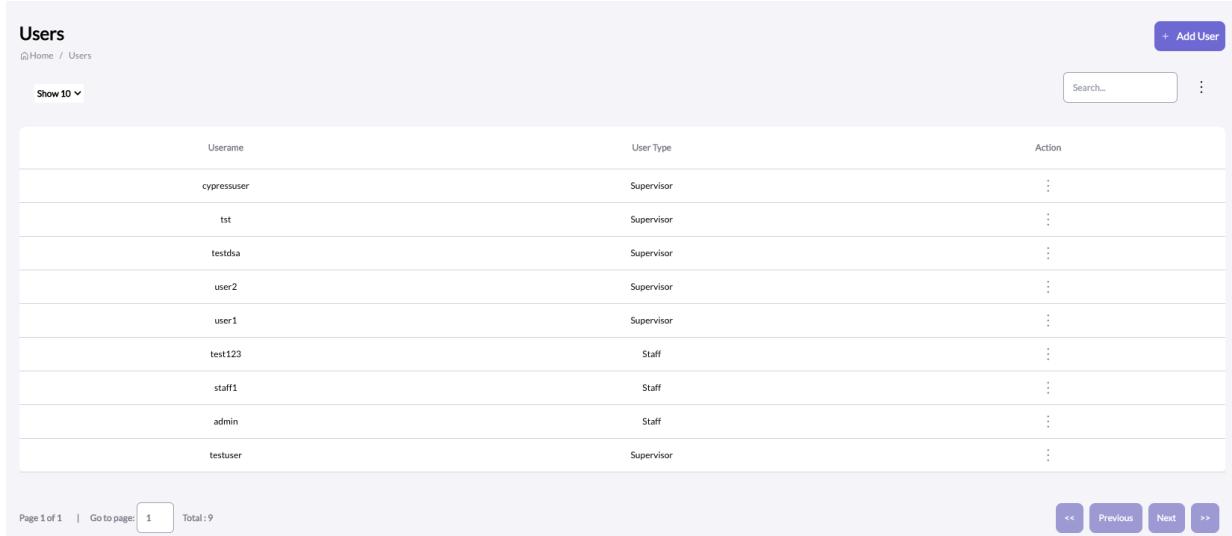
Figure 138: Screenshot of developed add user page

```

9  const AddUser = () => {
10    const navigate = useNavigate();
11    const notify = () => toast.success("User added successfully");
12    const notifyError = (msg) => toast.error(`${msg}`);
13    const [renderApp, setRenderApp] = useState(false)
14    const [userType, setUserType] = useState([
15      { value: 'supervisor', label: "Supervisor" },
16      { value: 'staff', label: "Staff" },
17    ])
18    const [selectedUserType, setSelectedUserType] = useState();
19    const [selectedRole, setSelectedRole] = useState();
20    const [permissions, setPermissions] = useState([]);
21
22    >
23    const { ... } = useForm();
24    const password = useRef();
25    password.current = watch('password', '');
26    const onSubmit = async (data) => {
27    };
28    const handleChildChange = (e, id) => {
29    }
30    const loadData = async () => {
31      try {
32        const response = await Promise.all([
33          axios.get("http://localhost:5100/api/permission-lists", { withCredentials: true }),
34        ])
35        setPermissions(response[0].data7.data)
36        setRenderApp(true)
37      } catch (err) {
38        console.log(err)
39      }
40    }
41    useEffect(() => {
42      loadData()
43    }, [])
44    return (
45      <div>
46        {renderApp &&
47         <div className='px-7 font-secondary py-10'>
48           <ToastContainer className='text-sm text-grey' />
49           <div className='font-bold text-2xl'>Add User</div>
50           <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
51             <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
52               <form onSubmit={handleSubmit(onSubmit)} action='-'>
53                 </form>
54               </div>
55             </div>
56           </div>
57         </div>
58       );
59     };
60     export default AddUser;
61

```

Figure 139: Screenshot of developed code for add user page.



The screenshot shows a table titled "Users" with the following data:

Username	User Type	Action
cypressuser	Supervisor	⋮
tst	Supervisor	⋮
testdsa	Supervisor	⋮
user2	Supervisor	⋮
user1	Supervisor	⋮
test123	Staff	⋮
staff1	Staff	⋮
admin	Staff	⋮
testuser	Supervisor	⋮

Below the table, there are navigation controls: "Page 1 of 1 | Go to page: 1 Total: 9" and buttons for "Previous" and "Next".

Figure 140: Screenshot of developed user page

```

6  import "react-toastify/dist/ReactToastify.css";
7  const User = () => {
8    const location = useLocation();
9    const [allUser, setAllUser] = useState([]);
10   const [renderApp, setRenderApp] = useState(false);
11  >   const loadData = async () => {-
12    };
13    useEffect(() => {
14      loadData();
15    }, [location]);
16
17  >   const notify = () => {-
18    ...;
19    const notifyError = (msg) => toast.error(`${msg}`);
20    const handleDelete = async (data) => {-
21    };
22    return (
23      <div className='px-7 font-secondary py-10'>
24        <ToastContainer className='text-sm text-grey' />
25        <div className='flex justify-between'>
26          <div className='>
27            <div className='font-bold text-2xl'>Users</div>
28            <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
29              <div className='first flex items-center'>
30                <span className='material-symbols-outlined text-sm'>home</span>
31                <span>Home</span>
32              </div>
33              <div className='divider mx-2'>/</div>
34              <div className='second'>Users</div>
35            </div>
36          </div>
37          <div className='flex'>
38            <Link
39              to='/users/add-user'
40              className='bg-primary text-white h-10 p-2 text-sm flex items-center font-main rounded'
41              >
42                <span className='material-symbols-outlined text-sm mr-2'>add</span>
43                <span>Add User</span>
44              </Link>
45            </div>
46          </div>
47        </div>
48        <div className='product-table'>
49          {renderApp && <Table handleDelete={handleDelete} data={allUser} />}
50        </div>
51      </div>
52    );
53  };
54
55  export default User;
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77

```

Figure 141: Screenshot of developed code for viewing user list.

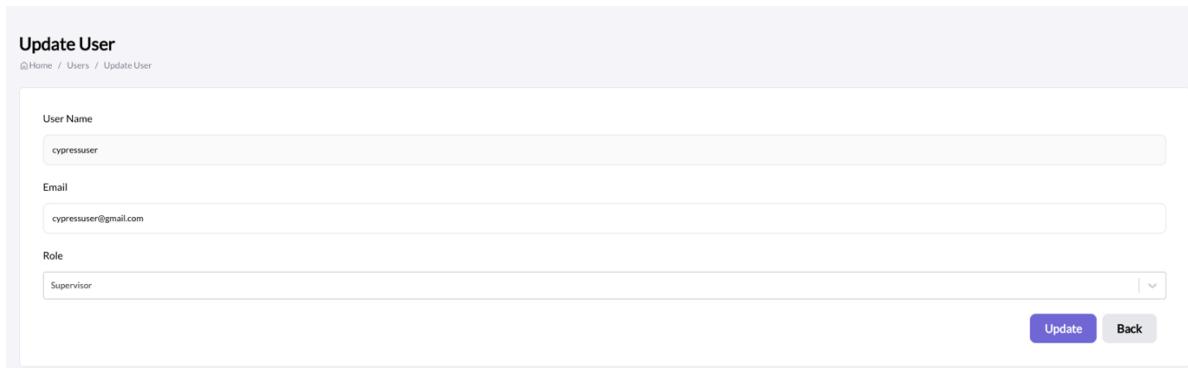


Figure 142: Screenshot of developed edit user page.

```

31 |   ...
32 | } = useForm();
33 | const [userType, setUserType] = useState([...])
34 |
35 | const onSubmit = async (data) => {
36 |   ...
37 | }
38 |
39 | const loadData = async () => {
40 |   try {
41 |     const response = await Promise.all([
42 |       axios.get("http://localhost:5100/api/permission-lists", { withCredentials: true }),
43 |       axios.get(`http://localhost:5100/api/user/${id}`, { withCredentials: true })
44 |     ])
45 |     setAvailableRole(response[0].data.data)
46 |     setPermissions(response[0].data.data)
47 |     setDbData(response[1].data.data[0])
48 |     console.log(response[1].data.data[0].permissions)
49 |     setSelectedRole(response[1].data.data[0].permissions.map(x => x.permission))
50 |     setSelectedUserType(userType.find(x => x.value === response[1].data.data[0].user_type))
51 |     setRenderApp(true)
52 |   } catch (err) {
53 |     console.log(err)
54 |   }
55 | }
56 |
57 | useEffect(() => {
58 |   loadData()
59 | }, [])
60 |
61 | const handleChildChange = (e, id) => {
62 |   ...
63 | }
64 |
65 | return (
66 |   <div>
67 |     {renderApp &&
68 |      <div className='px-7 font-secondary py-10'>
69 |        <ToastContainer className='text-sm text-grey' />
70 |        <div className='font-bold text-2xl'>Update User</div>
71 |        <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
72 |          <div className='first flex items-center'>
73 |            <span className='material-symbols-outlined text-sm'>home</span>
74 |            <span>Home</span>
75 |          </div>
76 |          <div className='divider mx-2'></div>
77 |          <div className='second'>Users</div>
78 |          <div className='divider mx-2'></div>
79 |          <div className='second'>Update User</div>
80 |        </div>
81 |        <div className='bg-white mt-5 p-8 rounded-md border font-thin'>
82 |          <form onSubmit={handleSubmit(onSubmit)} action='-'>
83 |            ...
84 |          </form>
85 |        </div>
86 |      </div>
87 |    );
88 |
89 | }
90 |
91 | </div>
92 |
93 | )
94 |
95 | )
96 |
97 | )
98 |
99 | )
100 |
101 |
102 |
103 |
104 |
105 |
106 |
107 |
108 |
109 |
110 |
111 |
112 |
113 |
114 |
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |
124 |
125 |
126 |
127 |
128 |
129 |
130 |
131 |
132 |
133 |
134 |
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |
143 |
144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |
168 |
169 |
170 |
171 |
172 |
173 |
174 |
175 |
176 |
177 |
178 |
179 |
180 |
181 |
182 |
183 |
184 |
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |

```

Figure 143: Screenshot of developed code for edit user page

3.7.5.3. Sprint Retrospective

In this sprint, the role-based system was successfully developed within the given time frame. However, changes in business logic caused some setbacks, and improvements could be made in terms of adaptability and stakeholder communication.

3.7.6. Sprint 6 – Search

3.7.6.1. Sprint Planning

In this sprint, the focus is on developing a fully functional and tested search feature for an application. This involves indexing all required data, adding new data to the Elastic search index, and creating a user interface for viewing search results. The sprint includes tasks such as developing Figma mockups, updating Elastic search indexes, ensuring consistency between databases, and building a searchable application.

3.7.6.2. Development

a) Figma Mockup

The Figma mockup displays a search results page titled "Search Result for 'B.N'" within a web-based application interface. The sidebar on the left lists navigation options: Dashboard, Customer, Products, and Invoice. The main content area shows three distinct sections: Customer, Products, and Employees, each listing items related to the search term "B.N".

Customer Section:

Name	Phone	Location
B.N Traders	5329953	Newroad, Kathmandu
B.N Suppliers	5329953	Newroad, Kathmandu

Products Section:

Name	Category	Price
B.N Socks	Ankle Socks	550

Employees Section:

Name	Phone	Salary
B.N Pandey	5329953	22000
B.N Pandey		22000

Figure 144: Figma mockup generated for search result.

b) Frontend code



Figure 145: Screenshot of developed search input field

```
69 |   const onSubmit = async (data) => {
70 |     navigate(`/search?q=${data.keyword}`);
71 |   }
72 |   return (
73 |     <>
74 |       {showNav && [
75 |         <div className='bg-white flex py-4 px-6'>
76 |           <div className='search-with-icons flex bg-background ms:ml-1 m'>
77 |             <i className='bi bi-search sm:block hidden text-grey'></i>
78 |             <form onSubmit={handleSubmit(onSubmit)}>
79 |               <input
80 |                 {...register('keyword')}
81 |                 id='search-keyword'
82 |                 className='bg-transparent sm:w-80 w-36 ml-2 rounded-lg'
83 |                 type='text'
84 |                 placeholder='Search...'
85 |               />
86 |             </form>
87 |           </div>
88 |         </div>
89 |       ]}</>
90 |     )
91 |   )
92 | }
```

Figure 146: Screenshot of developed code for search input field.

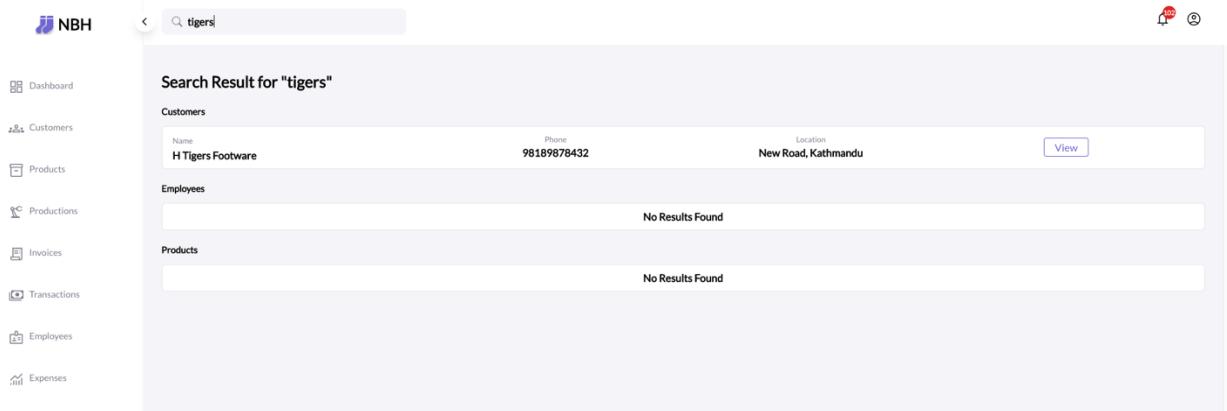


Figure 147: Screenshot of developed search result page

```

src > Pages > Search > Search.js > Search
  1 import axios from 'axios';
  2 import React, { useState, useEffect } from 'react';
  3 import { Link, useLocation } from 'react-router-dom';
  4 import { useAuth } from '../../../../../Components/Authentication/auth';
  5
  6 const Search = () => {
  7   const [searchKeyword, setSearchKeyword] = useState('');
  8   const [searchResult, setSearchResult] = useState();
  9   const [renderApp, setRenderApp] = useState(false)
 10  const { search } = useLocation();
 11  const auth = useAuth()
 12  const loadSearch = async () => {
 13    try {
 14      const response = await axios.post(`${auth?.baseURL}/api/search`, { keyword: `${searchKeyword}` })
 15      console.log(response?.data?.data)
 16      setSearchResult(response?.data?.data)
 17      setRenderApp(true)
 18    } catch (err) {
 19      console.log(err)
 20    }
 21  }
 22  useEffect(() => {
 23    loadSearch()
 24  }, [searchKeyword])
 25  useEffect(() => {
 26    const query = new URLSearchParams(search);
 27    const keyword = query.get('q') || '';
 28    setSearchKeyword(keyword);
 29  }, [search]);
 30  return [
 31    <div className='px-7 font-secondary py-10 relative'>
 32      <div className='!>
 33        <div className='font-bold text-2xl'>Search Result for <span id='search-keyword-view'>
 34          ${searchKeyword}</span>
 35        </div>
 36        {renderApp &&
 37          <div className='>
 38            <>...
 39            </>
 40            <>...
 41            </>
 42            <>...
 43              <div className='mt-5'>
 44                <div className='text-sm text-black text-thin font-secondary'>Employees</div>
 45              </div>
 46              {
 47                searchResult?.filter(x => x._index === "employees")?.length > 0
 48                ?
 49                  <>...
 50                  <div className='bg-white grid grid-cols-4 items-center content-center'>
 51                    <div className='>
 52                      <label className='text-xs text-grey'>
 53                        Name
 54                      </label>
 55                    </div>
 56                  </div>
 57                </>
 58              }
 59            </>
 60          </>
 61        </>
 62      </div>
 63    </div>
 64  ]
 65}
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
100
101
102
103

```

Figure 148: Screenshot of code developed for search result.

3.7.6.3. Testing

a) API Testing

```
(base) gaurav@Gauravs-MacBook-Pro backend % npm run test search.test.js
> test
> jest --runInBand --forceExit "search.test.js"

  PASS  __tests__/search.test.js
    POST /api/search
      ✓ Searches testing (90 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.892 s
```

Figure 149: Screenshot of test case result developed for Sprint 6

b) Cypress testing

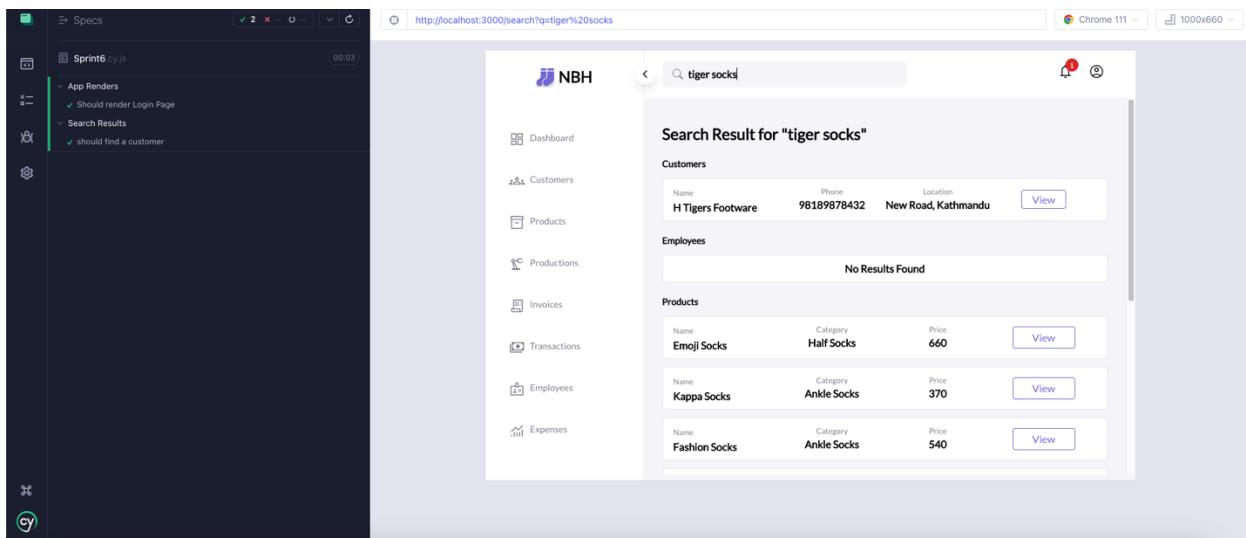


Figure 150: Successful test case for frontend components

3.7.6.4. Sprint Retrospective

In Sprint #06, Elasticsearch functionality was added to the system, and CRUD operations were performed to contribute data to the search index. The task was completed successfully, and an interim report was submitted. Planning for upcoming sprints could be improved by defining necessary activities and seeking expert assistance if documentation is difficult to understand. However, going back to modify working code was deemed inefficient.

3.7.7. Sprint 7 – Notifications

3.7.7.1. Sprint Planning

Sprint #07 is focused on improving the customer experience by implementing a notification system that will keep customers informed about their invoices. During the work on developing the code for sending notifications to customers when new invoices are generated, designing a user interface for viewing staff actions, and adding email notification functionality. The deliverables of the sprint include a Figma mockup, code for sending notifications, test plan and results documentation, and bug and issue resolution documentation.

a) Figma Mockup

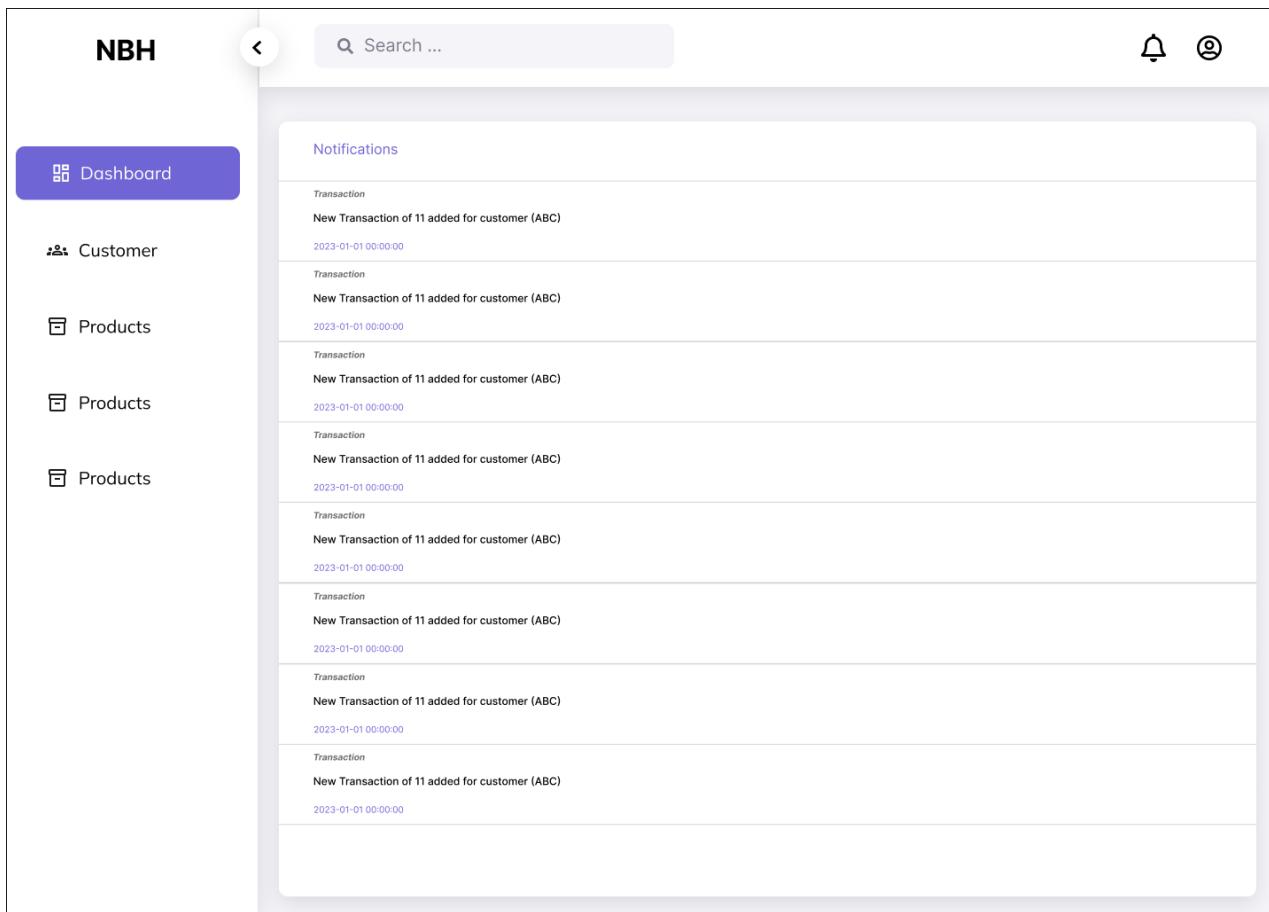


Figure 151: Figma mockup for view all notification page.

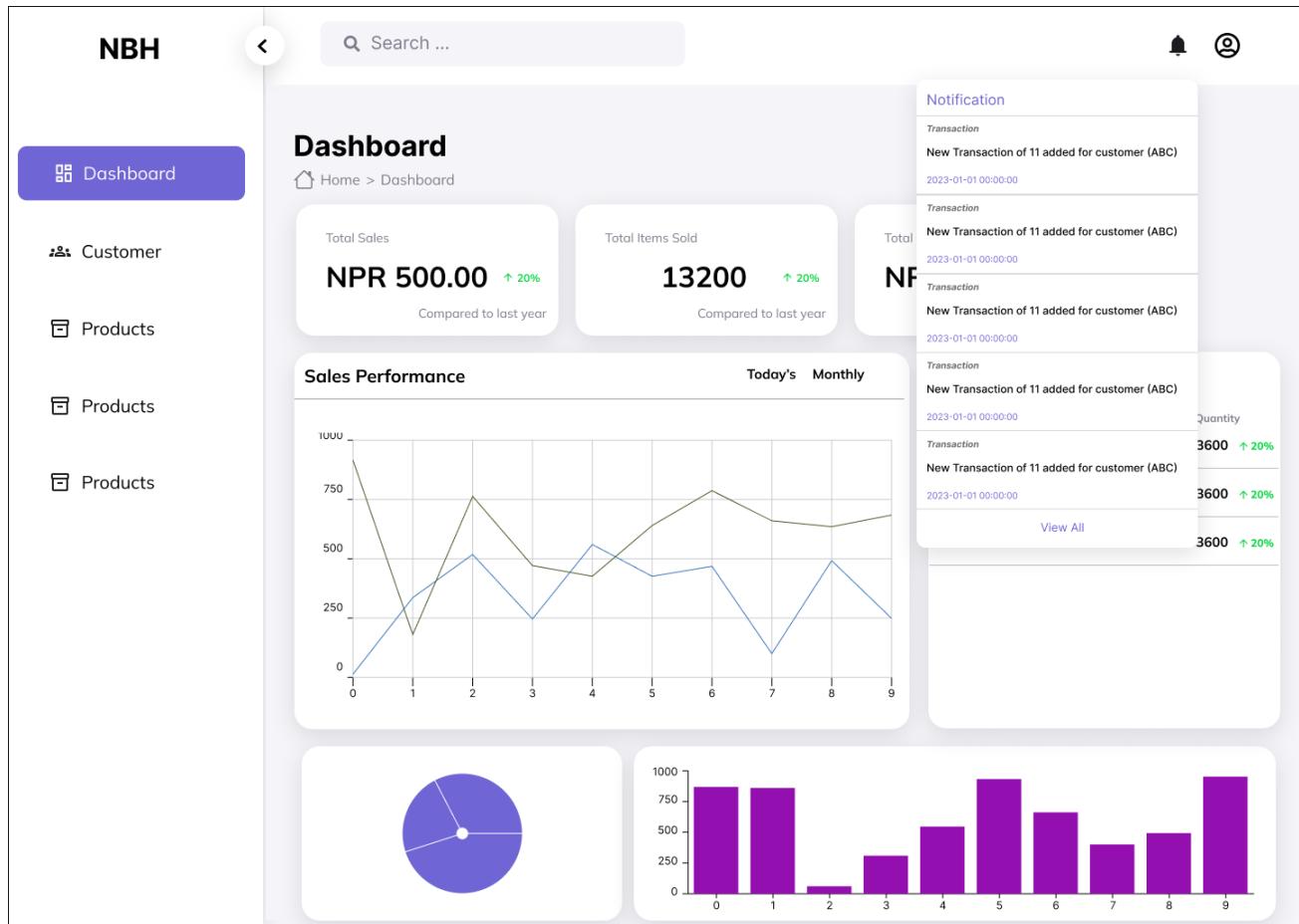


Figure 152: Figma mockup for preview of notification

3.7.7.2. Development

a) Table Design

TABLE NAME			
notifications			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity ◊	PRIMARY KEY +
user_type	text	no default ◊	+
notification_type	text	no default ◊	+
type	text	no default ◊	+
description	text	no default ◊	+
user_id	text	no default ◊	+
created_at	date	no default ◊	+

Indexes			
INDEX NAME	TYPE	COLUMNS	
notifications_pkey	Primary Key Index	id	

Figure 153: Table design for notification table

TABLE NAME			
user_notification_status			
COLUMN NAME	TYPE	DEFAULT	CONSTRAINTS
id	integer	identity ◊	PRIMARY KEY +
notification_id	integer	no default ◊	+
user_id	integer	no default ◊	+
status	text	constant ◊ UNREAD	status = ANY (ARRAY['READ'::text, 'UNREAD'::text]) +

Indexes			
INDEX NAME	TYPE	COLUMNS	
user_notification_status_pkey	Primary Key Index	id	

Figure 154: Table design for user notification status table

c) Frontend Development

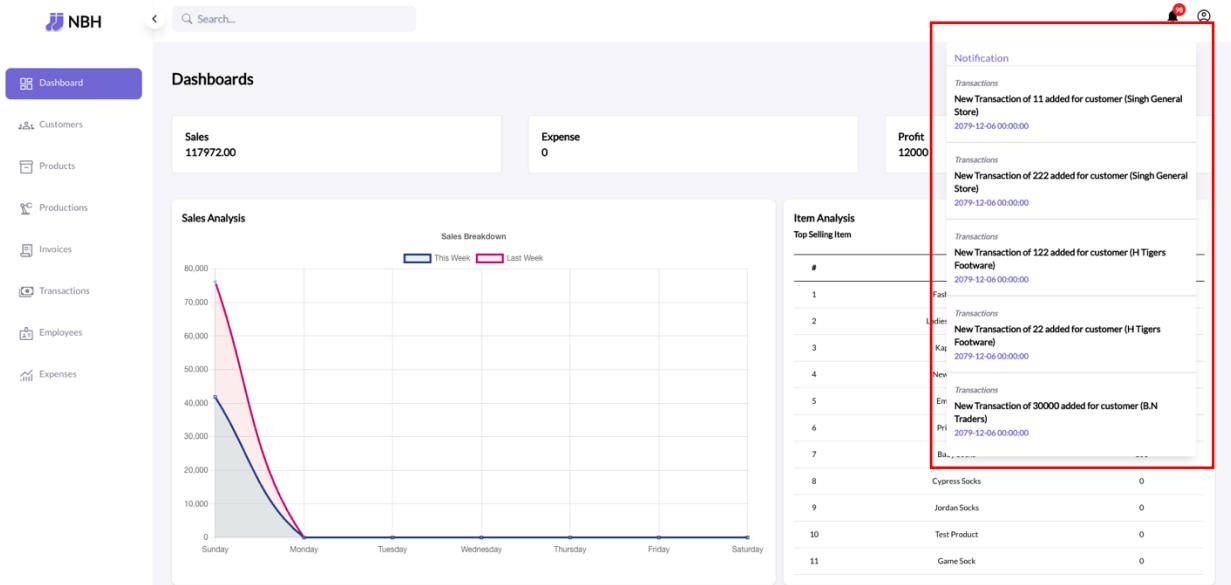


Figure 155: Screenshot of developed preview of notification

```
{showNav && (
  <div className='bg-white flex py-4 px-6'>
    <div className='search-with-icons flex bg-background ms:ml-1 ml-1 rounded-lg px-4 py-2'>
      <i className='bi bi-search sm:block hidden text-grey'></i>
      <input
        className='bg-transparent sm:w-80 w-36 ml-2 rounded-lg' ... />
    </div>
    <div className='ml-auto flex z-40'>
      {/* <span class='material-symbols-outlined mr-6'>notifications</span> */}
      <div ref={notificationRef} className="relative">
        <span ... />
        {notificationCount > 0 &&
          <div className="absolute -top-1 right-4 z-50 w-5 h-5 rounded-full bg-red-600 grid place-items-center text-[10px] text-white">{notificationCount}</div>}
        <div
          className={
            showNotification
              ? "absolute z-50 rounded shadow-lg py-3 sm:right-0 -right-8 top-14 sm:w-96 w-80 bg-white"
              : "absolute hidden rounded shadow-lg px-3 py-3 sm:right-0 -right-8 top-14 w-80 sm:w-96 bg-white"
          }
        >
          <div className="text-md text-primary font-medium px-3">
            Notification
          </div>
          {notification?.map((x, index) => (
            <div
              key={index}
              className="py-4 px-3 cursor-pointer text-sm hover:bg-blue-50 text-black first:border-t-0 border-y last:border-b-0" ...
            >
              <div className="!capitalize">
                <div className="italic text-[12px] mb-1 !capitalize text-grey">
                  {x.type}
                </div>
                <div className="">{x.description}</div>
                <div className="mt-1 text-primary text-xs">
                  {format(
                    new Date(x.created_at),
                    "yyyy-MM-dd HH:mm:ss"
                  )}
                </div>
              </div>
            )})
          </div>
        </div>
      </div>
    </div>
  </div>
)
```

Figure 156: Screenshot of developed code for notification preview

Notifications

Home / Notifications

Transactions
New Transaction of 11 added for customer (Singh General Store)
2079-12-06 00:00:00

Transactions
New Transaction of 222 added for customer (Singh General Store)
2079-12-06 00:00:00

Transactions
New Transaction of 122 added for customer (H Tigers Footware)
2079-12-06 00:00:00

Transactions
New Transaction of 22 added for customer (H Tigers Footware)
2079-12-06 00:00:00

Transactions
New Transaction of 30000 added for customer (B.N Traders)
2079-12-06 00:00:00

Transactions
New Transaction of 30000.00 added for customer (B.N Traders)
2079-12-07 00:00:00

Transactions
New Transaction of 30000.00 added for customer (B.N Traders)
2079-12-07 00:00:00

Transactions
New Transaction of 100000.00 added for customer (Singh General Store)
2079-12-07 00:00:00

Figure 157: Screenshot of developed all notification page.

```
src > Pages > Notification.jsx > [s] Notification
  6  const Notification = () => {
  7    const [allNotifications, setAllNotifications] = useState([])
  8    const [renderApp, setRenderApp] = useState(false)
  9    const auth = useAuth()
 10   const loadData = async () => {
 11     try {
 12       const response = await axios.get(`${auth?.baseUrl}/api/notifications`, {
 13         withCredentials: true
 14       })
 15       if (response.status === 200) {
 16         setRenderApp(true)
 17         setAllNotifications(response?.data?.data)
 18       }
 19     } catch (err) {
 20       console.log(err)
 21     }
 22   }
 23   useEffect(() => {
 24     loadData()
 25   }, [])
 26   return (
 27     <div className='px-7 font-secondary py-10'>
 28       <div className='flex justify-between'>
 29         <div>...
 30       </div>
 31       <div className='product-table mt-5 bg-white rounded'>
 32         {renderApp &&
 33           <div>
 34             {allNotifications.map((x, index) => (
 35               <div
 36                 key={index}
 37                 className="py-4 px-3 cursor-pointer text-sm hover:bg-blue-50 text-black first:border-t-0 border-y 1px border-gray-200">
 38                 <div className="!capitalize">
 39                   <div className="italic text-[12px] mb-1 !capitalize text-gray">
 40                     {x.type}
 41                   </div>
 42                 </div>
 43                 <div>{x.description}</div>
 44                 <div className="mt-1 text-primary text-xs">
 45                   {format(
 46                     new Date(x.created_at),
 47                     "yyyy-MM-dd HH:mm:ss"
 48                   )}
 49                 </div>
 50               </div>
 51             ))
 52           </div>
 53         )
 54       </div>
 55     </div>
 56   )
 57   </div>
 58   </div>
 59   </div>
 60   </div>
 61   </div>
 62   </div>
 63   </div>
 64   </div>
 65   </div>
 66   </div>
 67   </div>
 68 )
```

Figure 158: Screenshot of developed code for viewing all notifications.

3.7.7.3. Testing

a) API Testing

```
● (base) gaurav@Gauravs-MacBook-Pro backend % npm run test notification.test.js

> test
> jest --runInBand --force-exit "notification.test.js"

PASS __tests__/notification.test.js
  POST /api/notification
    ✓ Getting Notification for dashabord (14 ms)
    ✓ Getting All Notification (5 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.434 s
Ran all test suites matching /notification.test.js/i.
Force exiting test. Have you considered using `--detectOpenHandles` to detect an open browser instance?
```

Figure 159: Test case result developed for Sprint 7

b) Cypress Testing

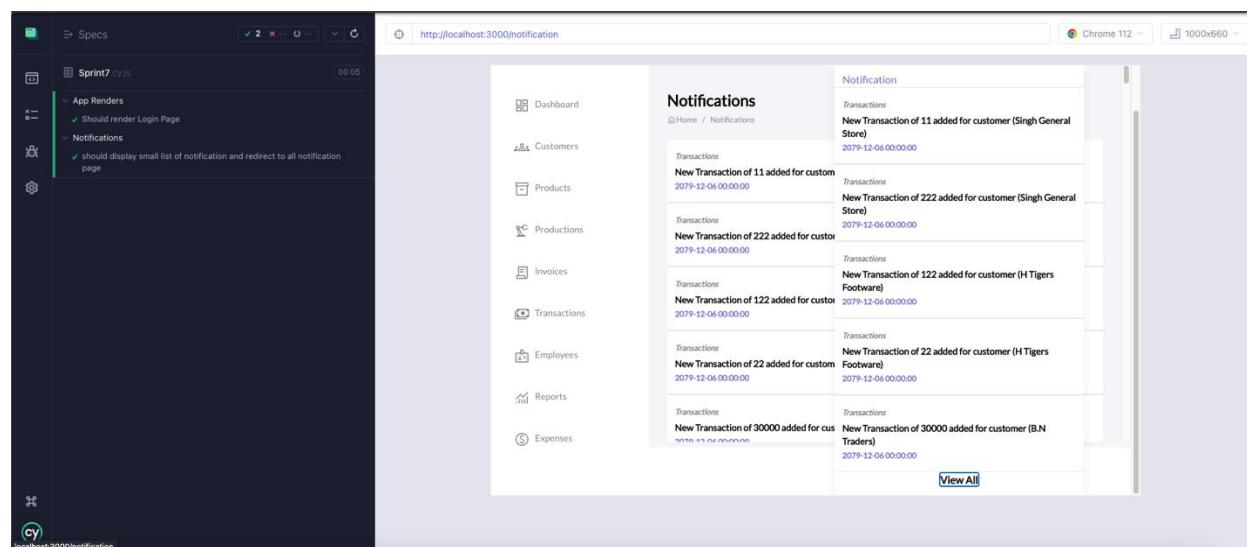


Figure 160: Successful test case for frontend components

3.7.7.4. Sprint Retrospective

The objective of this sprint was to implement notifications for various system activities, such as staff activity notifications on staff pages, customer email notifications when invoices are generated, and supervisor stock notifications sent via cron job. However, I encountered issues with email delivery due to mail blockage and formatting issues. To improve future sprints, more extensive testing and seeking professional help if needed can be considered. Overall, the sprint was a success in achieving the sprint goal.

3.7.8. Sprint 8 - QR Code Implementation

3.7.8.1. Sprint Planning

The goal of Sprint #08 is to add a new feature to the application that will allow users to insert details of products in packages and generate a QR code for the package that can be scanned later to add the products to invoices. The tasks for this sprint include developing all required Figma mockups, creating a functional feature for adding products to a package and generating a QR code, creating a QR code scanner to retrieve data stored in the QR.

a) Figma Mockup

The figure shows a Figma mockup of a web application interface for adding production. The header features the text "NBH" and a search bar labeled "Search ...". On the right side of the header are a bell icon and a user profile icon. The left sidebar contains navigation links: "Dashboard", "Customer", "Products", "Invoice", and "Production", with "Production" being the active tab. The main content area has a title "Add Production" and a breadcrumb trail "Home > Production > Add Production". A date field is set to "2079-10-11". Below this is a table with columns: "#", "Name", "Item Code", and "Quantity". A single row is present with the number "1" and a text input field "Type name here". To the right of the input field is a dropdown arrow and a small green square button. Below the table is a section labeled "Storage Room" with a text input field "Type room here ...". At the bottom right are two buttons: a purple "Add" button and a grey "Back" button.

Figure 161: Figma Mockup for adding production.

The screenshot shows a Figma mockup of a production management interface. On the left, a sidebar menu includes 'Dashboard', 'Customer', 'Products' (selected), and 'Productions'. The main area is titled 'Production' and shows a table with one row of data:

ID	Date	Availability	Storage Room	Action
ABCD	2079-10-12	Available	R001	⋮ View Production View QR

Below the table, there are navigation controls: 'Page 1 of 1 | Go to page: 1' and buttons for '<< Previous Next >>'. The top right features a search bar and icons for notifications and user profile.

Figure 162: Figma Mockup for production.

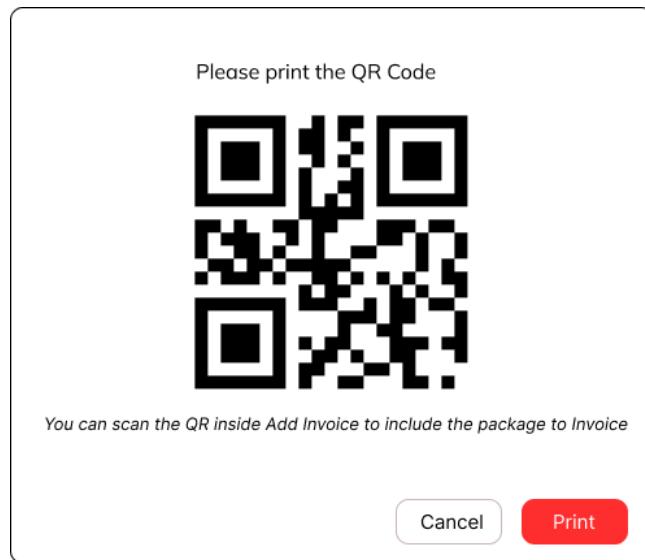
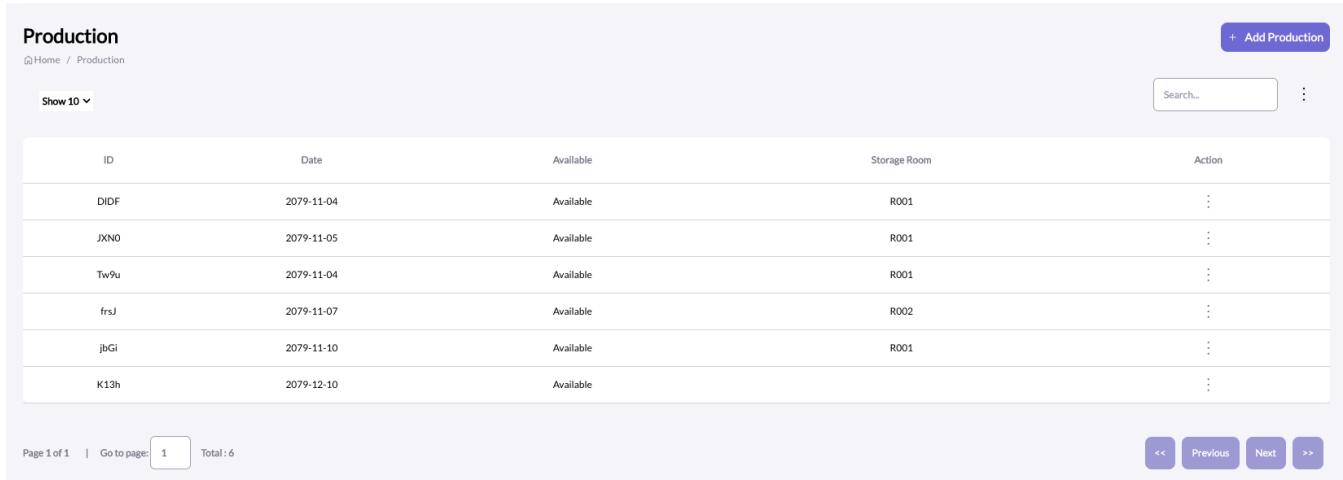


Figure 163: Figma mockup for QR code.

3.7.8.2. Development



The screenshot shows a web application interface for managing production. At the top right is a button labeled '+ Add Production'. Below it is a search bar with placeholder text 'Search...'. The main area displays a table with the following data:

ID	Date	Available	Storage Room	Action
DIDF	2079-11-04	Available	R001	⋮
JXNO	2079-11-05	Available	R001	⋮
Tw9u	2079-11-04	Available	R001	⋮
frsJ	2079-11-07	Available	R002	⋮
jbGi	2079-11-10	Available	R001	⋮
K13h	2079-12-10	Available		⋮

At the bottom left, there are navigation links: 'Page 1 of 1' and 'Go to page: 1'. To the right, it says 'Total: 6'. On the far right are buttons for navigation: '<<', 'Previous', 'Next', and '>>'.

Figure 164: Developed production list page

```
src > Pages > Production > Production.jsx > ...
8  const Production = () => {
9    const location = useLocation();
10   const [allProduction, setAllProduction] = useState([]);
11   const [renderApp, setRenderApp] = useState(false);
12   const loadData = async () => {
13     try {
14       const response = await axios.get(
15         "http://localhost:5109/api/production"
16       );
17       console.log(response);
18       response?.status === 200 && setAllProduction(response?.data);
19       setRenderApp(true);
20     } catch (err) {
21       console.log(err);
22     }
23   };
24   useEffect(() => {
25     [location];
26   }, [location]);
27
28   const notify = () => {
29     toast.success("Production deleted successfully");
30     loadData();
31   };
32   const notifyError = (msg) => toast.error(`$${msg}`);
33   const handleDelete = async (data) => {
34     ...
35
36     const handleEdit = async (data) => {};
37
38     return (
39       <div className='px-7 font-secondary py-10'>
40         <ToastContainer className='text-sm text-grey' />
41         <div className='flex justify-between'>
42           <div className='flex flex-grow'>
43             <div className='font-bold text-2xl'>Production</div>
44             <div className='breadcrumb-container flex items-center mt-1 text-xs font-thin text-grey'>
45               <div className='first flex items-center'>
46                 <span className='material-symbols-outlined text-sm'>home</span>
47                 <span>Home</span>
48               </div>
49               <div className='divider mx-2'></div>
50               <div className='second'>Production</div>
51             </div>
52           </div>
53           <div className='flex-grow'>
54             <Link
55               to='/production/add-production'
56               className='bg-primary text-white p-2 text-sm flex items-center font-main rounded-lg'>
57               <span className='material-symbols-outlined text-sm mr-2'>add</span>
58               <span>Add Production</span>
59             </Link>
60           </div>
61         </div>
62       </div>
63       <div className='product-table'>
64         {renderApp && <Table handleDelete={handleDelete} data={allProduction} />}
65       </div>
66     
```

Figure 165: Developed code for viewing production.

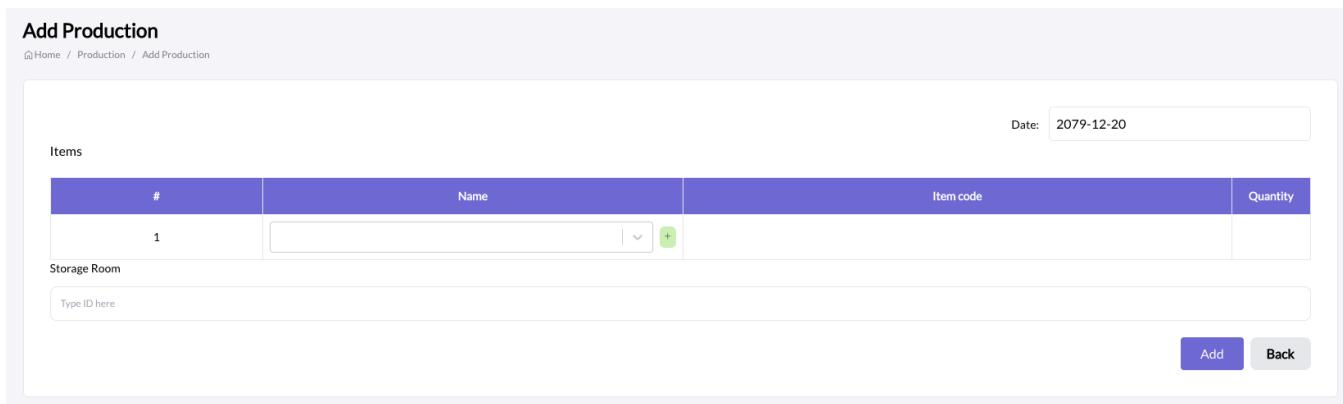


Figure 166: Developed add production page.

```

src > Pages > Production > AddProduction.jsx > [e] AddProduction
  ...
14 const AddProduction = () => [
15   const { id } = useParams();
16   const [productView, setProductView] = useState(false);
17   const [renderApp, setRenderApp] = useState(false);
18   const [date, setDate] = useState(adToBs(format(Date.now(), "yyyy-MM-dd")));
19   const [productList, setProductList] = useState([]);
20   const [navigate] = useNavigate();
21   const [waitingResponse, setWaitingResponse] = useState(false);
22   ...
23   const handleDate = ({ bsDate, adDate }) => {
24     ...
25   };
26   ...
27   const [items, setItems] = useState([
28     ...
29   ]);
30   ...
31   const addFields = (index, value, event) => {
32     ...
33   };
34   ...
35   const [form, { errors }] = useForm();
36   const notify = () => toast.success("Invoice added successfully");
37   ...
38   const loadData = async () => {
39     ...
40   };
41   ...
42   useEffect(() => {
43     ...
44   }, [id]);
45   const submitData = async (data) => {
46     ...
47   };
48   const onSubmit = async (data) => {
49     ...
50     ...
51     ...
52     ...
53     ...
54     ...
55     ...
56     ...
57     ...
58     ...
59     ...
60     ...
61     ...
62     ...
63     ...
64     ...
65     ...
66     ...
67     ...
68     ...
69     ...
70     ...
71     ...
72     ...
73     ...
74     ...
75     ...
76     ...
77     ...
78     ...
79     ...
80     ...
81     ...
82     ...
83     ...
84     ...
85     ...
86     ...
87     ...
88     ...
89     ...
90     ...
91     ...
92     ...
93     ...
94     ...
95     ...
96     ...
97     ...
98     ...
99     ...
100    ...
101    ...
102    ...
103    ...
104    ...
105    ...
106    ...
107    ...
108    ...
109    ...
110    ...
111    ...
112    ...
113    ...
114    ...
115    ...
116    ...
117    ...
118    ...
119    ...
120    ...
121    ...
122    ...
123    ...
124    ...
125    ...
126    ...
127    ...
128    ...
129    ...
130    ...
  
```

Figure 167: Developed code for add production.

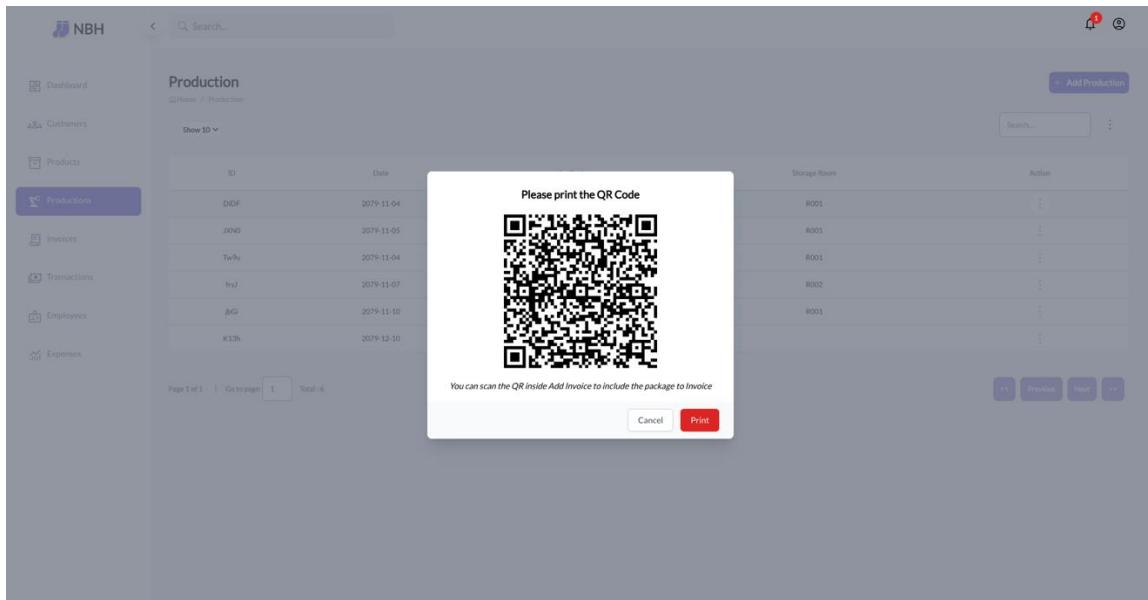


Figure 168: Developed QR code modal.

```

<div className="relative z-10 aria-labelledby="modal-title" role="dialog" aria-modal="true">
  <div className="fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity"></div>

  <div className="fixed inset-0 z-10 overflow-y-auto">
    <div className="flex min-h-full items-end justify-center p-4 text-center sm:items-center sm:p-0">
      <div className="relative transform overflow-hidden rounded-lg bg-white text-left shadow-xl transition-all sm:my-8 sm:w-full sm:max-w-lg">
        <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">
          {
            isLoading ?
              <>
                <div className="flex flex-col items-center">
                  <div className="mb-4 text-center font-bold">
                    Generating QR Code
                  </div>
                  <div role="status" class="space-y-8 animate-pulse md:space-y-0 md:space-x-8 md:flex md:items-center">
                    <div class="flex items-center justify-center w-full h-48 bg-gray-50 rounded sm:w-96 dark:bg-gray-300">
                      ...
                    </div>
                    <span class="sr-only">Generating QR Code...</span>
                  </div>
                </div>
              </> :
              <>
                {qrData &&
                  <div className="flex flex-col items-center">
                    <div className="text-lg font-bold mb-5">Please print the QR Code</div>
                    <div id="printable-content">
                      <QRCode value={'${qrData}'} />
                    </div>
                    <div className="text-sm text-center italic mt-5">You can scan the QR inside Add Invoice to include the package to Invoice</div>
                  </div>
                }
              </>
            }
          </div>
        </div>
        <div className="bg-gray-50 px-4 py-3 sm:flex sm:flex-row-reverse sm:px-6">
          <button onClick={(e) => { e.preventDefault(); handlePrint() }} type="button" className="inline-flex w-full justify-center rounded-md border border-gray-300 sm:mr-3">
            Print
          </button>
          <button onClick={() => { setModal(false) }} type="button" className="mt-3 inline-flex w-full justify-center rounded-md border border-gray-300">
            Cancel
          </button>
        </div>
      </div>
    </div>
  </div>
</div>

```

Figure 169: Developed code for QR Code modal

3.7.8.3. Sprint Retrospective

The QR implementation sprint was a success, where project successfully converting JSON data from backend to QR code in frontend. However, the project encountered a major setback when Chrome restricted use of the camera and microphone on HTTP sites, which made it difficult for them to do the jobs as intended. The project can improve future sprints by understanding the problem not only from the code level but as the project manager of the project, and by seeking help from the professional if any error is messing with the timeline of the project.

3.7.9. Sprint 9 – Import / Export Files

3.7.9.1. Sprint Planning

The goal of Sprint #9 is to add a bulk import and export feature for CSV files, as well as the ability to download data in PDF and CSV formats. Tasks include developing Figma mockups, creating a functional feature for importing and exporting data, and adding verification and validation for importing data. Deliverables include a fully tested and functional application capable of importing data from CSV files with appropriate messaging, as well as a Figma-designed mockup for importing from CSV.

a) Figma Mockup

The figure shows a Figma mockup of a web application interface. At the top left is a logo 'NBH'. A search bar with placeholder 'Search ...' is at the top center. Top right icons include a bell and a user profile. On the left is a sidebar with 'Dashboard', 'Customer', 'Products', 'Invoice', and 'Employee' buttons; 'Employee' is highlighted with a purple background. The main content area has a title 'Employee' and a breadcrumb 'Home > Employee'. It shows two rows of data in a table:

ID	Name	Location	Phone
1	001	B.N Traders	22222
1	001	B.N Traders	22222

To the right of the table is a context menu with checkboxes for 'Location' and 'Phone', and options to 'Download PDF', 'Download CSV', or 'Upload CSV'. At the bottom are navigation links: 'Page 1 of 1 | Go to page: 1', '<<', 'Previous', 'Next', and '>>'.

Figure 170: Figma Mockup for menu design for downloading.

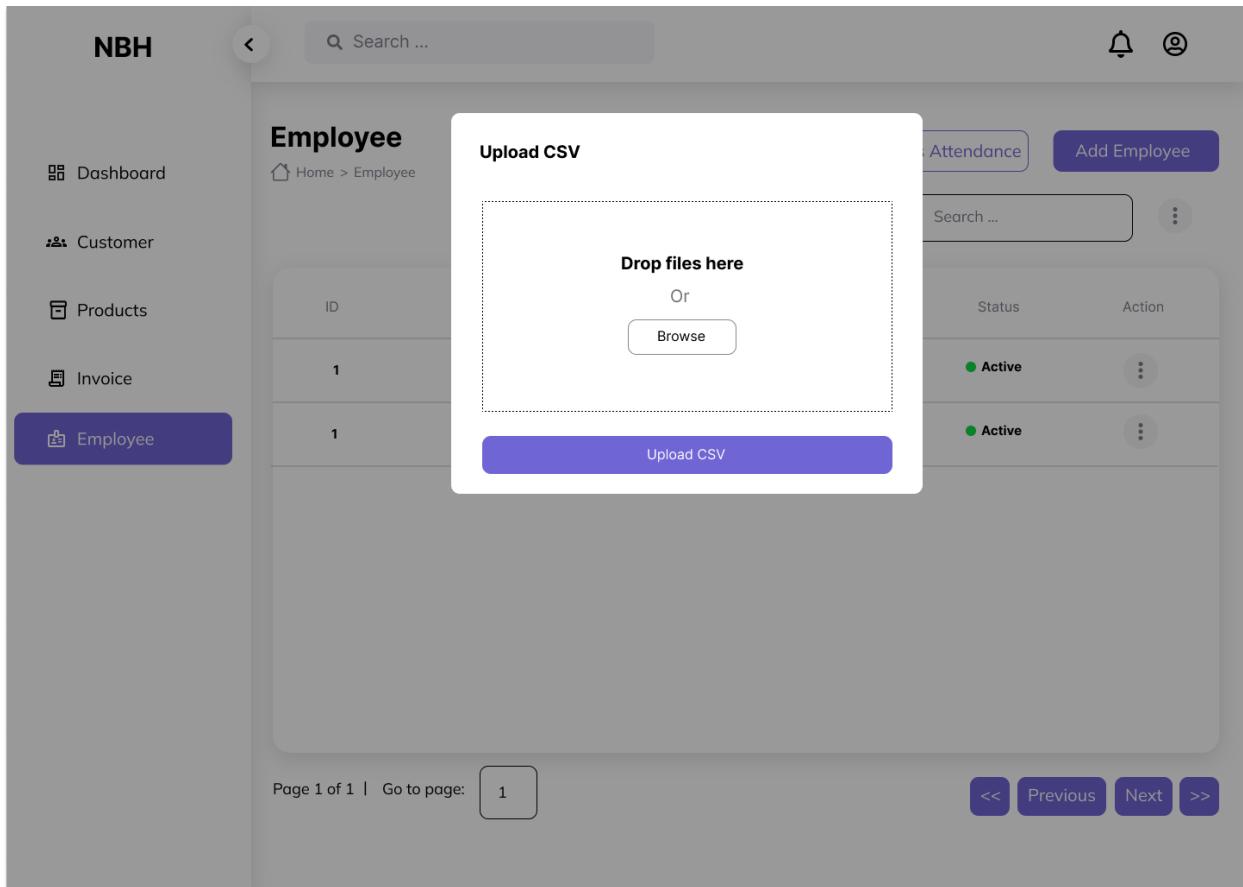


Figure 171: Figma Mockup for uploading CSV file.

3.7.9.2. Development

a) Frontend Development

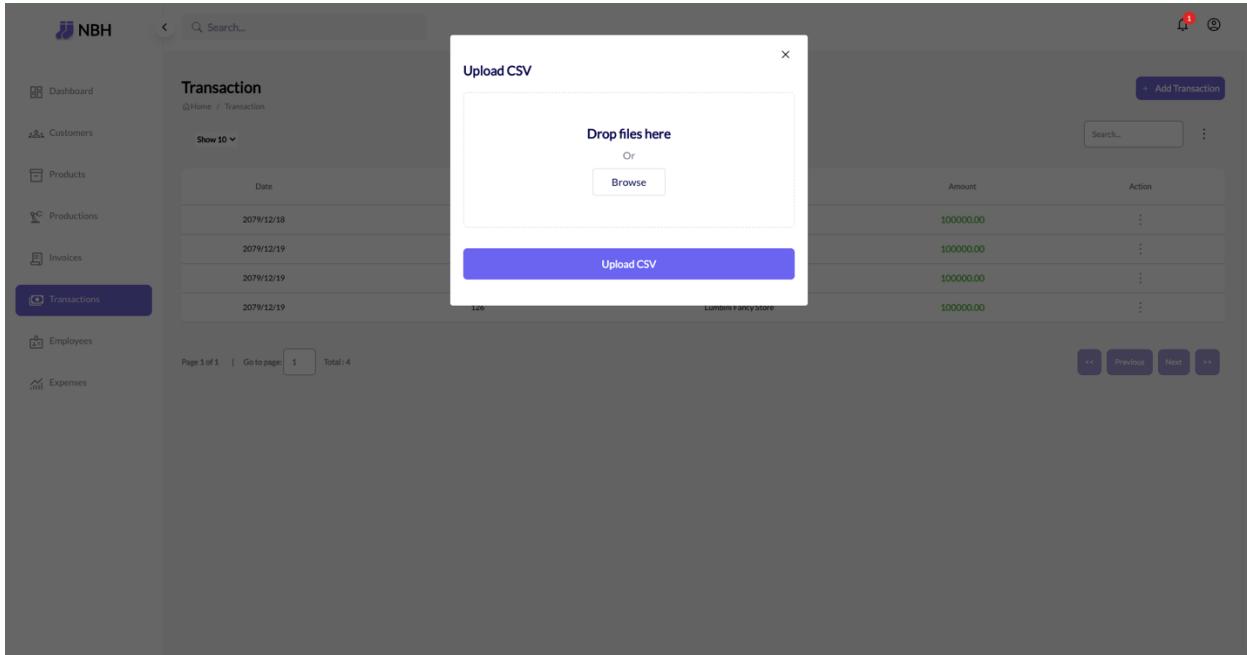


Figure 172: Developed CSV upload component

```

src > Components > ImportModal.jsx > ImportModal > submitFiles
  4   import { useState } from 'react';
  5   const ImportModal = ({ setModal, errMessage, handleSubmit, successMessage, inserting, loadData }) => {
  6 >     const ... =
  7     } = useFileUpload();
  8     const submitFiles = () => [
  9       Papa.parse(files[0], {
 10         header: true,
 11         skipEmptyLines: true,
 12         complete: function (results) {
 13           handleSubmit(results.data)
 14         },
 15       });
 16     ];
 17     const inputRef = useRef();
 18     return (
 19       <div className='fixed min-h-[100vh] min-w-[100vw] top-0 right-0 bg-black bg-opacity-60'>
 20         <div className='flex relative items-center justify-center p-12'>
 21           <div className='mx-auto relative w-full max-w-[550px] bg-white px-5 py-10 rounded'>
 22             <button onClick={()=> {
 23               if (files.length > 0 &&
 24                 loadData());
 25             }
 26             setModal(false); clearAllFiles()
 27           }>
 28             <span className='material-symbols-outlined hover:bg-slate-200 text-white absolute top-4 right-5'>
 29               close
 30             </span>
 31           </button>
 32           <form>
 33             >
 34               <div className='mb-6'>
 35                 <label className='mb-5 block text-xl font-semibold text-[#07074D]>
 36                   Upload CSV
 37                 </label>
 38
 39               <div className='mb-8'>...
 40                 <div>
 41                   {fileNames.map((name) => ...
 42                     ))
 43                   <>...
 44                   </>
 45
 46                   </div>
 47
 48                 </div>...
 49               </div>
 50             </div>
 51           </form>
 52         </div>
 53       </div>
 54     );
 55   }
 56   export default ImportModal

```

Figure 173: Code developed for uploading CSV.

The screenshot shows a web-based application interface for managing transactions. At the top, there's a navigation bar with 'Home / Transaction' and a button '+ Add Transaction'. Below the navigation is a search bar labeled 'Search...'. A dropdown menu is open on the right side, containing the following options:

- Date
- Transaction ID
- Party Name
- Amount
- Download PDF
- Download CSV
- Upload CSV

Below the search bar, there's a 'Show 10' dropdown. The main area displays a table of transactions with the following data:

Date	Transaction ID	Party Name	Amount
2079/12/18	122	H Tigers Footware	100000.00
2079/12/19	124	Lumbini Fancy Store	100000.00
2079/12/19	125	Lumbini Fancy Store	100000.00
2079/12/19	126	Lumbini Fancy Store	100000.00

At the bottom left, there are page navigation controls: 'Page 1 of 1', 'Go to page: 1', and 'Total: 4'. On the far right, there are buttons for '<<', 'Previous', 'Next', and '>>'.

Figure 174: Developed download file menu

3.7.9.3. Testing

a) Cypress Testing

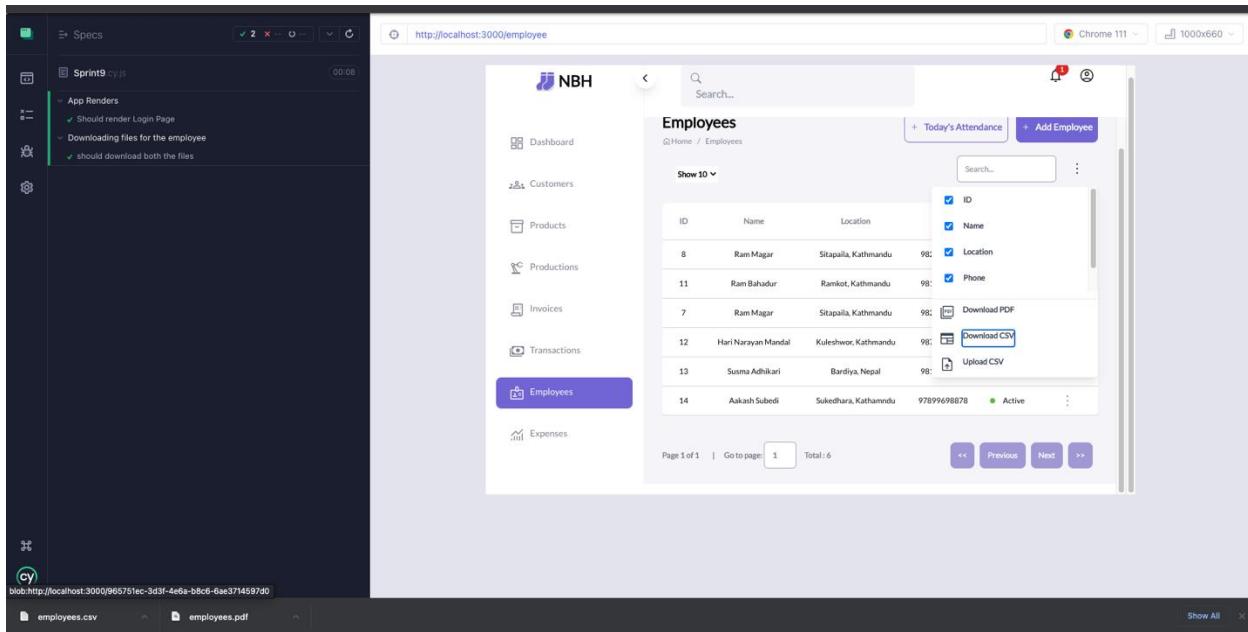


Figure 175: Successful test case for frontend components

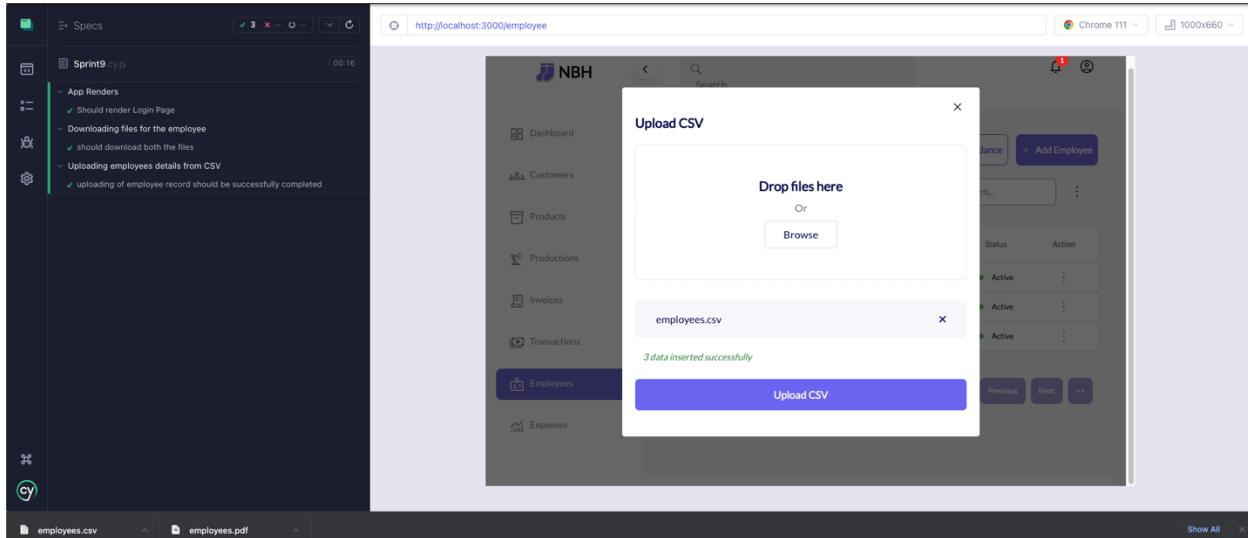


Figure 176: Successful test case for frontend components

3.7.9.4. Sprint Retrospective

The sprint goal was to create a feature for exporting data in CSV, PDF, and XLSX formats, but only CSV and PDF formats were completed due to poor time management and other academic project assignments. To improve future sprints, the individual plans to create a new Gantt chart including setbacks, move unfinished tasks to the next sprint, and maintain efficient communication with the client and supervisor.

3.7.10. Sprint 10 – Report Generation

3.7.10.1. Sprint Planning

In Sprint #10, the goal is to create APIs for supervisor, admin, and customer views of the dashboard, and to create different charts to visualize data from these APIs. Tasks include developing Figma mockups, creating APIs and charts, and fully testing the application. Deliverables include a fully tested and functional application capable of generating different reports and Figma-designed mockups for the dashboard and reports.

a) Figma Mockup

The Figma mockup displays the Admin Dashboard interface. On the left sidebar, there are three items: 'Dashboard' (highlighted in purple), 'Users', and 'Recent Cron History'. The main content area has a header 'Dashboard' and a breadcrumb 'Home > Dashboard'. It features three summary cards: 'Total Users' (4), 'Total Supervisor' (1), and 'Total Staff' (3). Below these is a table titled 'Recent Cron History' with columns: ID, Date, Description, Status, and Remarks. The table contains two entries:

ID	Date	Description	Status	Remarks
001	2079-11-20	Holiday Attendance	Success	-
002	2079-11-21	Stock Alert Mail Send	Success	-

Figure 177: Figma Mockup for admin dashboard

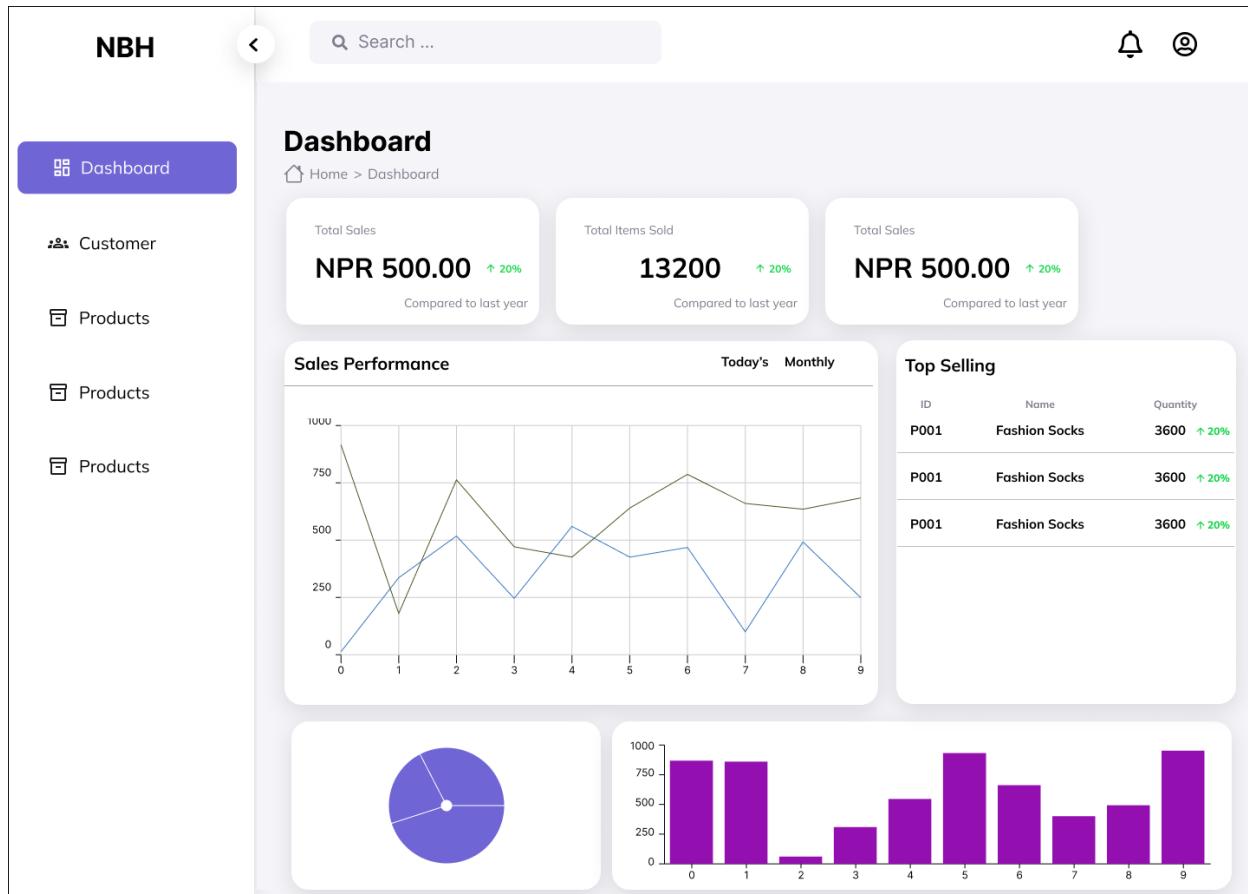


Figure 178: Figma Mockup for Supervisor dashboard

3.7.10.2. Development

a) Frontend Development

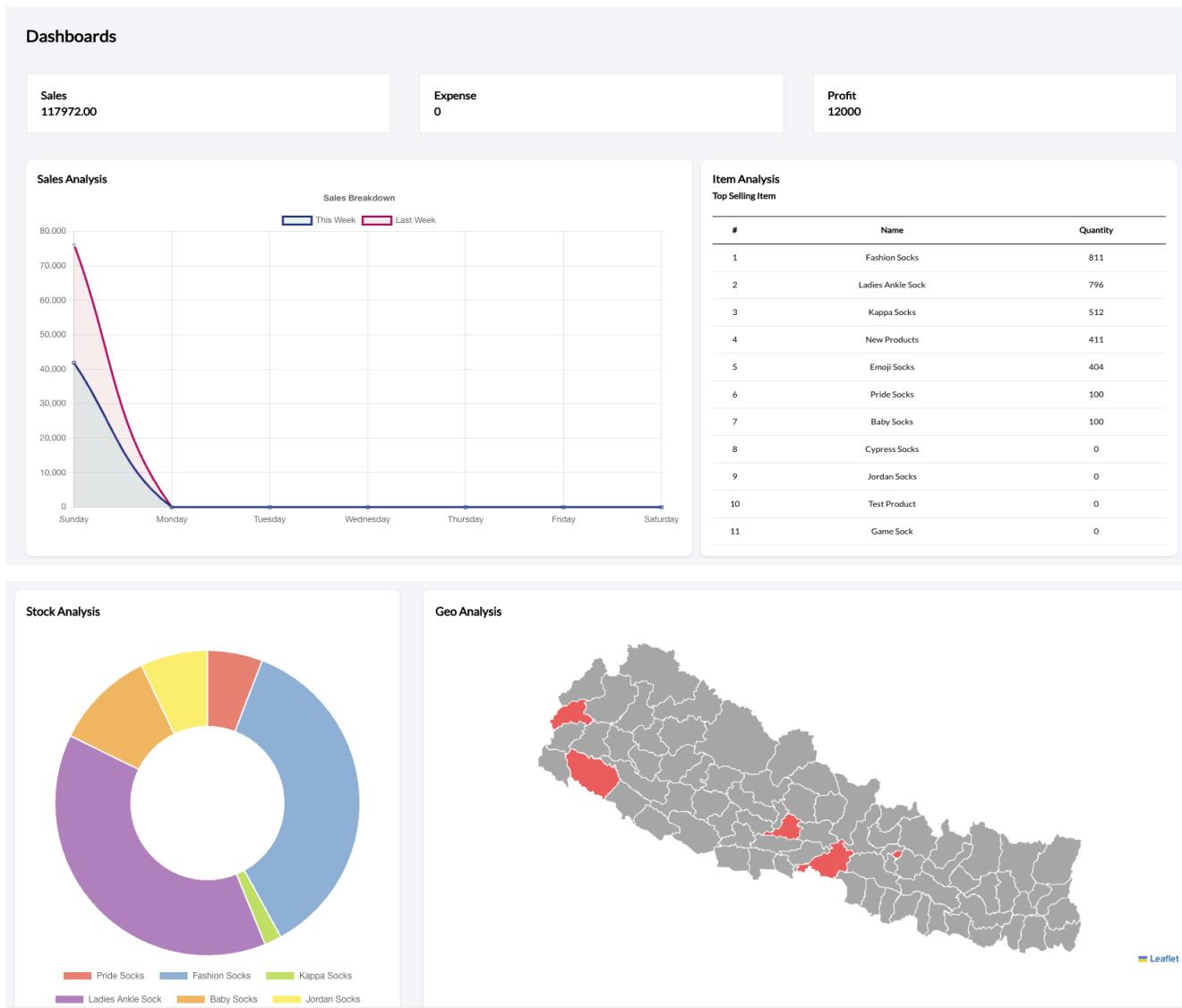


Figure 179: Developed supervisor dashboard

```
src > Pages > Dashboard > js Dashboard.js > ...
  ...
  34 |   ...
  35 |   const Dashboard = () => {
  36 |     const [data, setData] = useState()
  37 |     const options = { ... }
  38 |   ...
  39 |   const [dashboard, setDashboard] = useState()
  40 |   const [renderApp, setRenderApp] = useState(false)
  41 |   const [selectedFeature, setSelectedFeature] = useState(null)
  42 |
  43 |   const handleFeatureHover = (properties) => { ... }
  44 |
  45 |   const auth = useAuth()
  46 |   const loadData = async () => { ... }
  47 |
  48 |
  49 |
  50 |
  51 |
  52 |   const doughnutOpitons = { ... }
  53 |
  54 |
  55 |
  56 |
  57 |
  58 |
  59 |
  60 |
  61 |
  62 |
  63 |
  64 |
  65 |
  66 |
  67 |   useEffect(() => {
  68 |     loadData()
  69 |   }, [])
  70 |
  71 |   const doughnutLabel=
  72 |
  73 |   const [donughtdata, setDoughtData] = useState()
  74 |   const [geoMapData, setGeoMapData] = useState()
  75 |
  76 |
  77 |
  78 |
  79 |
  80 |
  81 |
  82 |
  83 |
  84 |
  85 |
  86 |
  87 |
  88 |   // const doughnutLabel=
  89 |
  90 |   const [donughtdata, setDoughtData] = useState()
  91 |   const [geoMapData, setGeoMapData] = useState()
  92 |
  93 |
  94 |   useEffect(() => { ... })
  95 |   [dashboard])
  96 |   return (
  97 |     <div className='px-7 font-secondary py-10'>
  98 |       <div className='font-bold text-2xl'>Dashboards</div>
  99 |       {renderApp &&
 100 |         <div>
 101 |           <div className=' mt-10'>
 102 |             <div className='grid grid-cols-1 sm:grid-cols-3 gap-3 sm:gap-10'>
 103 |               <div className='border bg-white p-5 rounded'>
 104 |                 <span className='text-black font-bold font-main'>...</span>
 105 |               </div>
 106 |               <div>
 107 |                 | {dashboard?.sales?.thisWeek}
 108 |               </div>
 109 |             </div>
 110 |           <div className='border bg-white p-5 rounded'>...
 111 |             </div>
 112 |           <div className='border bg-white p-5 rounded'>...
 113 |             </div>
 114 |           </div>
 115 |         </div>
 116 |       </div>
 117 |       <div className='sm:flex gap-3'>...
 118 |         </div>
 119 |       <div className='flex w-full h-full'>...
 120 |     </div>
```

Figure 180: Code developed for supervisor dashboard.

3.7.10.3. Testing

a) API Testing

```
> test
> jest --runInBand --forceExit "reports.test.js"

PASS  __tests__/reports.test.js
  GET /api/dashabord
    ✓ Getting Supervisor Dashboard Data (23 ms)
    ✓ Getting Admin Dashboard Data (5 ms)
    ✓ API validation (2 ms)
    ✓ Report Generation request (4 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.797 s, estimated 1 s
Ran all test suites matching /reports.test.js/i.
```

Figure 181: Test case result developed for Sprint 10

b) Cypress Testing

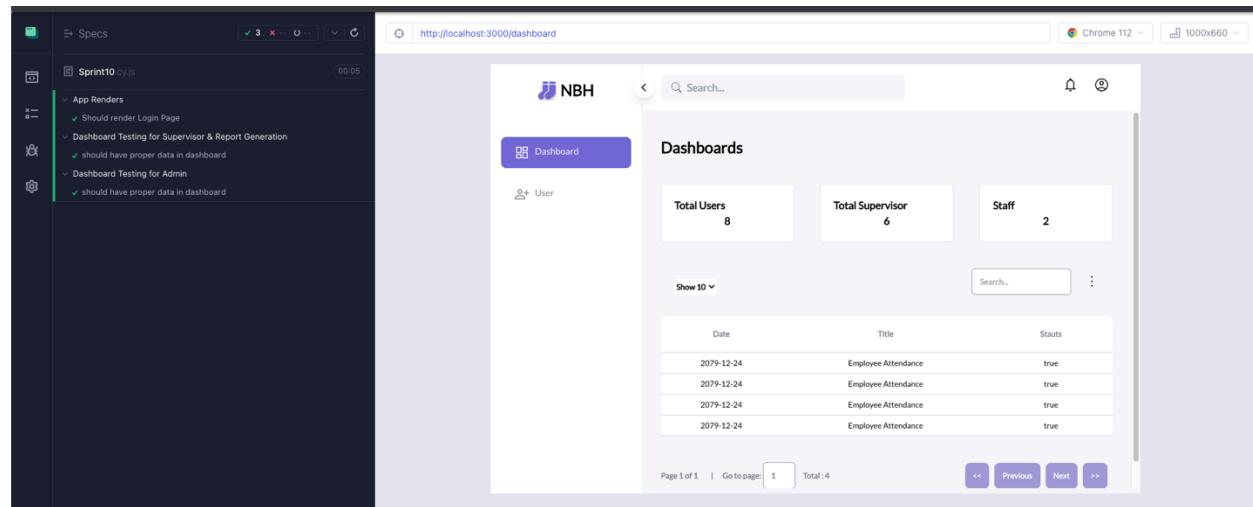


Figure 182: Successful test case for frontend components

3.7.10.4. Sprint Retrospective

The goal of Sprint #10 was to create reports and finish the dashboard content, which was achieved successfully. To improve future sprints, the individual plans to evaluate and improve their time management strategy, modify their planning and estimation procedure, review professional coding style, and maintain efficient communication with the client and supervisor to ensure everyone is aware of their obligations and deadlines.

4. Testing and analysis

As the automatic test cases for the application are developed and included in the implementation section, this section would conduct manual testing for each feature of the application.

4.1. Testing

4.1.1. Test 1 - Login Testing with Invalid login credentials

Test	1
Description	Login in the system with invalid login credentials
Action	Passing invalid login credentials in the login form username: test password: testeds
Expected Result	Error message should be displayed stating invalid username or password
Actual Result	Error message was displayed stating invalid username or password
Conclusion	Success

Table 20: Login Page with invalid credentials

The screenshot shows a 'Login' page with a red box highlighting an error message: 'Username or password is incorrect'. The 'Username' field contains 'test' and the 'Password' field contains 'testteds'. There is a 'Remember me' checkbox and a 'Login' button.

Figure 183: Error message for logging with invalid credentials

4.1.2. Test 2 - Login in the system with valid login credentials

Test	2
Description	Login in the system with valid login credentials
Action	Passing valid login credentials in the login form username: test password: test
Expected Result	User should be given success message and should be redirected to respective dashboard.
Actual Result	User was given success message and was redirected to respective dashboard.
Conclusion	Success

Table 21: Login Page with valid credentials

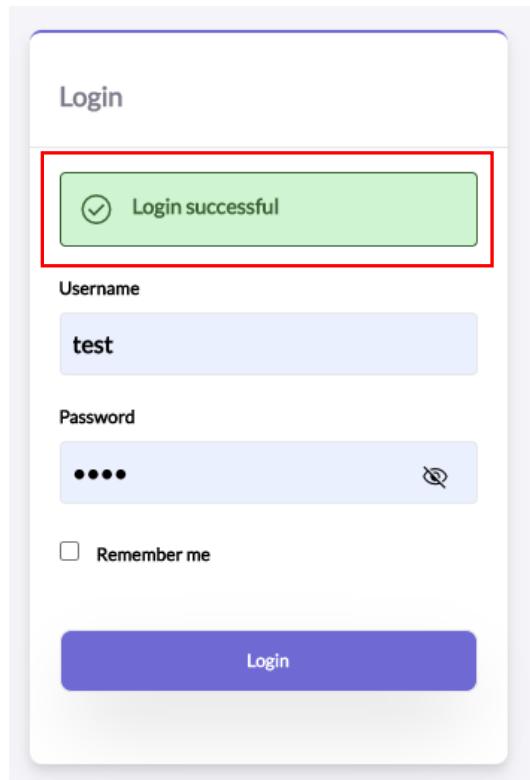


Figure 184: Success message for logging with valid credentials

The screenshot shows a web-based application interface. On the left is a sidebar with various menu items: Dashboard, Customers, Products, Productions, Invoices, Transactions, Employees, and Expenses. The "Dashboard" item is currently selected, indicated by a blue background. The main content area has a header "Dashboards". Below the header are three summary boxes: "Sales 143962.00", "Expense 0", and "Profit 23400". To the right of these boxes is a chart titled "Sales Analysis" showing "Sales Breakdown" for the week. The chart has two lines: a blue line for "This Week" and a red line for "Last Week". The x-axis shows days from Sunday to Saturday, and the y-axis shows sales volume from 0 to 80,000. A legend at the top of the chart indicates "This Week" (blue) and "Last Week" (red). To the right of the chart is a table titled "Item Analysis Top Selling Item". The table lists nine items with their names and quantities:

#	Name	Quantity
1	Fashion Socks	811
2	Ladies Ankle Sock	796
3	Kappa Socks	412
4	New Products	411
5	Emoji Socks	404
6	Pride Socks	100
7	Baby Socks	100
8	Jordan Socks	0
9	Game Sock	0

Figure 185: User being redirected to respective dashboard.

4.1.3. Test 3 - Testing form for possibilities of SQL injection

Test	3
Description	Testing form for possibilities of SQL injection
Action	Passing string closing in username. username: test'; SELECT * FROM users. password: test
Expected Result	Error message should be displayed stating no account on that username
Actual Result	The application crashed and displayed error message in console.
Conclusion	Failed

Table 22: Testing Login Form for SQL injection

Result:

```

error: syntax error at or near "''"
at Parser.parseErrorMessage (/Users/gaurav/Desktop/NBH/backend/node_modules/pg-protocol/dist/parser.js:287:98)
at Parser.handlePacket (/Users/gaurav/Desktop/NBH/backend/node_modules/pg-protocol/dist/parser.js:126:29)
at Parser.parse (/Users/gaurav/Desktop/NBH/backend/node_modules/pg-protocol/dist/parser.js:39:38)
at Socket.<anonymous> (/Users/gaurav/Desktop/NBH/backend/node_modules/pg-protocol/dist/index.js:11:42)
at Socket.emit (node:events:527:28)
at addChunk (node:internal/streams/readable:315:12)
at readableAddChunk (node:internal/streams/readable:289:9)
at Socket.Readable.push (node:internal/streams/readable:228:10)
at TCP.onStreamRead (node:internal/stream_base_commons:190:23) {
  length: 92,
  severity: 'ERROR',
  code: '42601',
  detail: undefined,
  hint: undefined,
  position: '65',
  internalPosition: undefined,
  internalQuery: undefined,
  where: undefined,
  schema: undefined,
  table: undefined,
  column: undefined
}

```

Figure 186: Error message in console.

4.1.4. Test 4 - Adding new user with existing username.

Test	4
Description	Adding new user with existing username
Action	Passing new user details with existing username
Expected Result	User should be giving error message with cause of error.
Actual Result	User should be giving error message with cause of error.
Conclusion	Success

Table 23: Testing new user registration with existing username.

Result:

The screenshot shows the NBH application's user management interface. On the left, there's a sidebar with icons for Dashboard, User (highlighted in purple), and Settings. The main area has a header with the NBH logo, a search bar, and navigation links for Home, Users, and Add User. Below this is a form titled 'Add User' with fields for User Name (containing 'test'), Email (containing 'admin@gmail.com'), Password (containing '*****'), and Confirm Password (containing '*****'). A dropdown for User Type is set to 'Supervisor'. At the bottom right are 'Add' and 'Back' buttons. An error message 'The username is already used' is displayed below the User Name field.

Figure 187: Error message on existing username.

4.1.5. Test 5 - Adding new user with valid details.

Test	5
Description	Adding new user with valid details
Action	Passing new user details with valid details
Expected Result	User should be given success toast message and redirect to user's page
Actual Result	User was given success toast message and redirect to user's page
Conclusion	Success

Table 24: Testing new user registration with valid details.

Result:

The screenshot shows a user interface for adding a new user. At the top, there is a header 'Add User' and a breadcrumb navigation 'Home / Users / Add User'. Below this is a form with four input fields: 'User Name' containing 'testuser', 'Email' containing 'testuser@gmail.com', 'Password' containing '*****', and 'Confirm Password' also containing '*****'. To the right of the form, a green success toast message 'User added successfully' is displayed.

Figure 188: User being added successfully.

4.1.6. Test 6 - API testing for protected routes.

Test	6
Description	Testing the protected route
Action	Requesting the list of products without logging in.
Expected Result	Error message should be displayed for unauthorized access
Actual Result	Error message was displayed for unauthorized access
Conclusion	Success

Table 25: Testing to access protected route without login.

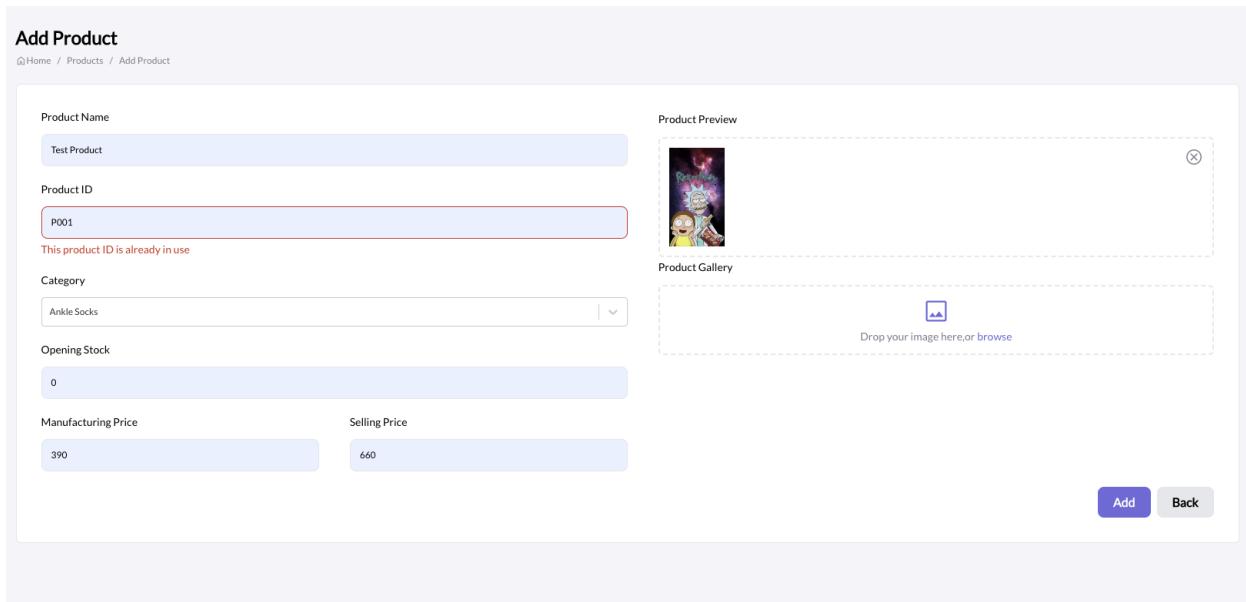
The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:5100/api/products`. Below the URL, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Params tab is selected, showing a table for Query Params with one row: Key (Value) and Value (Description). In the main body area, there are tabs for Body, Cookies, Headers (9), and Test Results. The Body tab is selected, showing a JSON response with three lines of code. The first line is a blank line, the second line starts with '1 | msg:', and the third line starts with '2 | "msg": "Unauthorized Access! Please Login to continue"'. Above the JSON response, the status bar indicates: Status: 401 Unauthorized Time: 53 ms Size: 354 B Save Response. Below the JSON response, there are buttons for Pretty, Raw, Preview, Visualize, and JSON, along with a search icon.

Figure 189: Error message thrown for unauthorized access.

4.1.7. Test 7 - Testing to add products pages with existing product id.\

Test	7
Description	Testing to add products pages with existing product id
Action	Passing below product data to the form. product name: Test Product product id: P001 opening stock: 0 manufacturing price: 360 selling price: 600
Expected Result	Error message should be displayed stating duplicate data entry
Actual Result	Error message was displayed stating duplicate data entry
Conclusion	Success

Table 26: Testing to add product with duplicate id.



The screenshot shows a web-based application for adding a new product. The page title is "Add Product". The URL in the address bar is "Home / Products / Add Product".

The form fields are as follows:

- Product Name:** Test Product
- Product ID:** P001 (This field has a red border, indicating an error.)
- This product ID is already in use** (A red error message displayed below the ID field.)
- Category:** Ankle Socks
- Opening Stock:** 0
- Manufacturing Price:** 390
- Selling Price:** 600

On the right side of the form, there are two sections:

- Product Preview:** Shows a small thumbnail image of a cartoon character.
- Product Gallery:** A placeholder area with a "Drop your image here, or browse" instruction and a file icon.

At the bottom right of the form are two buttons: "Add" (in blue) and "Back" (in grey).

Figure 190: Error message thrown for existing id.

4.1.8. Test 8 - Testing to add products pages with valid details.

Test	8
Description	Testing to add products pages with valid details
Action	Passing below product data to the form. product name: Test Product product id: P0013 opening stock: 0 manufacturing price: 360 selling price: 600
Expected Result	Success message should be displayed stating user about product being added.
Actual Result	Success message was displayed stating user about product being added.
Conclusion	Success

Table 27: Testing for adding a new product with valid details.

Add Product

Product Name: Test Product

Product ID: P014

Category: Ankle Socks

Opening Stock: 0

Manufacturing Price: 390

Selling Price: 660

Product Preview:

Product Gallery: Drop your image here or browse

Success message: Product added successfully

Figure 191: Success message displayed for adding new product.

4.1.9. Checking whether location of the customer update on selecting customer.

Test	9
Action	Description: Selecting a customer from a select.
Excepted Result	The address box form should update on selecting customer
Achieved Result	The address box form updated on selecting customer
Conclusion	Success

Table 28: Testing whether location updates based on the customer.

The screenshot shows the 'Add Invoice' interface. At the top left is a search bar with placeholder text 'Search...'. Below it is the title 'Add Invoice' and a breadcrumb trail 'Home / Invoice / Add Invoice'. The main area has a 'Customer Name' section with a dropdown menu. A red box highlights this dropdown, which lists several customers: 'Singh General Store', 'B.N Traders', 'Lumbini Fancy Store', 'H Tigers Footware', and 'AB Store'. The 'AB Store' option is currently selected. To the right of the dropdown is a date field set to '2079-11-23'. Below the dropdown is a table header for entering items, followed by a single row with a '#', 'Name' column containing an empty input field, and other columns for 'Item code', 'Quantity', 'Price', and 'Amount'. At the bottom right are 'Add' and 'Back' buttons.

Figure 192: Choosing a customer from dropdown.

This screenshot shows the same 'Add Invoice' interface as Figure 192, but with a different focus. A red box highlights the 'Location' input field, which now contains the text 'Birjung, Nepal'. The rest of the interface is identical to Figure 192, including the customer selection dropdown, date field, item table, and buttons at the bottom.

Figure 193: Location changed after choosing a customer.

4.1.10. Test 10 - Adding empty invoice.

Test	10
Description	Testing to add empty invoice to the application
Action	Passing below product data to the form. Customer: AB Store
Expected Result	Error message should be displayed with validation error
Actual Result	Error message was displayed with validation error
Conclusion	Success

Table 29: Testing to add empty invoice.

The screenshot shows the 'Add Invoice' interface. At the top, there's a navigation bar with 'Home / Invoice / Add Invoice'. Below it, the form fields are filled: 'Customer Name' is set to 'AB Store', 'Location' is 'Birjung, Nepal', and the date is '2079-12-17'. A red box highlights a tooltip-like message in the top right corner that says 'Empty Invoice cannot be created'. The 'Items' section contains a single row with a '#', 'Name' (empty), 'Item code' (empty), 'Quantity' (empty), 'Price' (empty), and 'Amount' (empty). The total value is '0'. There are 'Add' and 'Back' buttons at the bottom.

Figure 194: Error message being displayed.

4.1.11. Test 11 - Changing Employee Status

Test	11
Description	Testing to change employee status
Action	Sending archive request with employee id
Expected Result	Success message should be sent from server.
Actual Result	Success message was sent from server.
Conclusion	Success

Table 30: Testing to change employee status.

The screenshot shows a POST request to the endpoint `((URL))/api/archive/employee/7`. The 'Params' tab is selected, showing a single parameter 'Key' with a value of 'Value'. The 'Body' tab is selected, showing a JSON response:

```

1  {
2      "success": true,
3      "message": "Archived Successfully"
4  }

```

The status bar at the bottom right indicates `Status: 200 OK`.

Figure 195: Employee status being successfully updated.

4.1.12. Test 12 - Checking frontend validation for adding employee.

Test	12
Description	Checking frontend validation for adding employee
Action	Passing empty data and submitting the form
Expected Result	Error message should be displayed with validation error
Actual Result	Error message was displayed with validation error
Conclusion	Success

Table 31: Testing form validation for adding employees.

The screenshot shows a web-based application for adding an employee. The page title is 'Add Employee'. The URL in the address bar is 'Home / Employees / Add Employee'. The form contains several input fields:

- Employee Name:** A text input field with placeholder 'Type name here' is highlighted with a red border, indicating it is a required field. Below the field, the error message 'This field is required' is displayed in red.
- Phone:** A text input field with placeholder 'Type phone number here' is highlighted with a red border, indicating it is a required field. Below the field, the error message 'This field is required' is displayed in red.
- PAN Number:** A text input field with placeholder 'Type PAN here' is shown without a red border.
- Location:** A text input field with placeholder 'Type address here' is highlighted with a red border, indicating it is a required field. Below the field, the error message 'This field is required' is displayed in red.
- Salary:** A text input field with placeholder 'Type salary here' is highlighted with a red border, indicating it is a required field. Below the field, the error message 'This field is required' is displayed in red.
- Joined Date:** A date input field with the value '2079-12-18' is shown without a red border.

At the bottom right of the form, there are two buttons: a blue 'Add' button and a grey 'Back' button.

Figure 196: Error message being displayed.

4.1.13. Test 13 - Adding employee attendance.

Test	13
Description	Adding employee attendance for all active employee
Action	Passing attendance for all employee for with present and absent status respectively
Expected Result	The attendance should be added successfully with success message.
Actual Result	The attendance was added successfully with success message.
Conclusion	Success

Table 32: Testing for adding employee attendance.

The screenshot shows a user interface for managing employee attendance. At the top, there's a navigation bar with 'Home / Employees / Attendance'. Below it, a success message 'Attendance recorded successfully' is shown in a toast notification. The main area contains a table with columns for '#', Name, and Action. Two rows are present: one for 'Aashish Rai' marked as 'Present' and another for 'Binod Thapa Magar' also marked as 'Present'. A date input field shows '2079-12-18'. At the bottom, there are buttons for 'Add' and 'Back'.

#	Name	Action
1	Aashish Rai	Present
2	Binod Thapa Magar	Present

Figure 197: Screenshot of success message for adding attendance.

4.1.14. Test 14 - Adding duplicate attendance entry of all employees.

Test	14
Description	Adding duplicate attendance entry of all employees.
Action	Passing attendance for all employee for already existing attendance.
Expected Result	Error message should be displayed with warning of duplicate attendance entry.
Actual Result	Error message was displayed with warning of duplicate attendance entry.
Conclusion	Success

Table 33: Testing for adding duplicate entry for attendance.

The screenshot shows a web-based application for managing employee attendance. At the top, there's a navigation bar with links for Home, Employees, and Attendance. The main title is "Attendance". Below the title, there's a breadcrumb trail: Home / Employees / Attendance. A red box highlights a modal dialog box in the upper right corner. The dialog contains a red exclamation mark icon and the text "Attendance for same day cannot be added". Above the table, there's a date input field with the value "2079-12-18" also enclosed in a red box. The main content area displays a table of attendance entries:

#	Name	Action
1	Aashish Rai	Present
2	Binod Thapa Magar	Present

Below the table, there's a "Mark All:" section with radio buttons for "Present" (selected) and "Absent". At the bottom right are two buttons: "Add" and "Back".

Figure 198: Screenshot of error message being displayed for duplicate entry

4.1.15. Test 15 - Adding new employee record.

Test	15
Description	Adding new employee record.
Action	Passing below details for adding new employees. employee name: Ram Bahadur phone: 9812221123 location: <u>Ramkot</u> , Kathmandu salary: 22000 joined date: 2079-12-18
Expected Result	Success message should be displayed for adding new employee
Actual Result	Success message was displayed for adding new employee
Conclusion	Success

Table 34: Testing for adding new employee.

The screenshot shows the NBH application interface. On the left, there's a sidebar with icons for Dashboard, Customers, Products, Productions, Invoices, Transactions, Employees (which is highlighted in purple), and Expenses. The main area has a search bar at the top. Below it, a modal window titled "Add Employee" is open. The modal contains fields for Employee Name (Ram Bahadur), Phone (9812221123), Location (Ramkot, Kathmandu), Salary (22000), and Joined Date (2079-12-18). To the right of the location field is a placeholder "Type PAN here". At the bottom of the modal are "Add" and "Back" buttons. A success message "Employee added successfully" with a checkmark icon is displayed in a toast notification at the top right of the screen.

Figure 199: Screenshot of employee being added to the application.

4.1.16. Test 16 - Testing the functionality of data being inserted in elastic search indexes.

Test	16
Description	Testing the functionality of data being inserted in elastic search indexes.
Action	Adding new employee and checking the elastic search for the insert data.
Expected Result	Added data should be inserted in the employee index of the database.
Actual Result	Added data was inserted in the employee index of the database.
Conclusion	Success

Table 35: Testing to ensure data entry in elastic search database.

Add Employee

Employee Name: Babin Magar

Phone: 9812289662

PAN Number: 984029091

Location: Nuwakot, Nepal

Salary: 28000

Joined Date: 2079-12-21

Employee added successfully

Figure 200: Screenshot of adding employees.

```

1 - [
2   "_index": "employees",
3   "_id": "21",
4   "_version": 1,
5   "_seq_no": 33,
6   "_primary_term": 30,
7   "found": true,
8   "_source": {
9     "name": "Babin Magar",
10    "phone": "9812289662",
11    "location": "Nuwakot, Nepal",
12    "pan": "984029091",
13    "joined_date": "2079-12-21",
14    "is_active": "active"
15  }
16 ]

```

Figure 201: Screenshot of Kibana response for searching added employees.

4.1.17. Test 17 - API testing for searching data.

Test	17
Description	API testing for searching data.
Action	Passing a search keyword as: 'Binod Tha'
Expected Result	All data related to the search keyword should be returned in the response
Actual Result	All data related to the search keyword was returned in the response
Conclusion	Success

Table 36: Testing for created search query.

```

FYP / Sprint 6 - Search / Search
POST http://localhost:5100/api/search
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1
2 "keyword": "Binod Tha"
3

Body Cookies (1) Headers (10) Test Results
Pretty Raw Preview Visualize JSON
1
2 "data": [
3   {
4     "_index": "employees",
5     "_id": "9",
6     "_score": 20.415867,
7     "_source": {
8       "name": "Binod Thapa Magar",
9       "phone": "9812221222",
10      "location": "fas",
11      "pan": "604326666",
12      "joined_date": "2079-11-09T18:15:00.000Z",
13      "is_active": "active"
14    }
15  ]
16
17

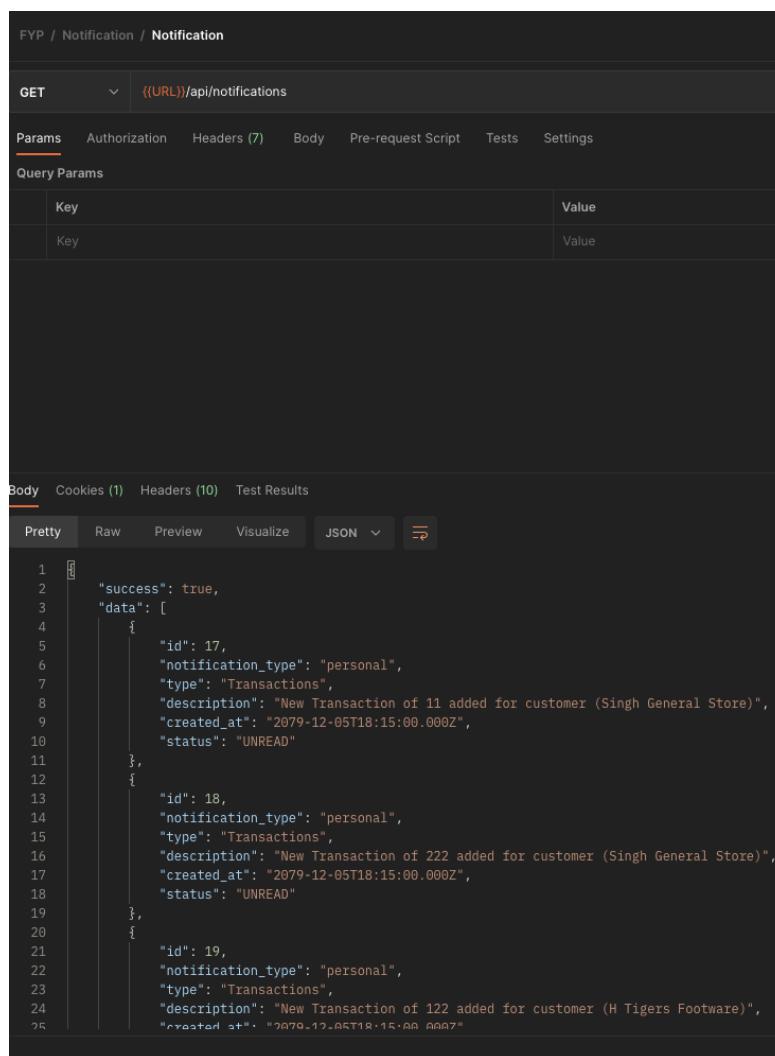
```

Figure 202: Screenshot of search result for requested keyword.

4.1.18. Test 18 - API testing for getting all user notifications.

Test	18
Description	API testing for getting all user notifications.
Action	Sending API request for notifications
Expected Result	A list of notification should be received in response.
Actual Result	A list of notification was received in response.
Conclusion	Success

Table 37:Testing to get all notification for a specific user.



The screenshot shows a Postman API test configuration for 'Notification / Notification'. The method is 'GET' and the URL is '({URL})/api/notifications'. The 'Params' tab is selected, showing a table for 'Query Params' with one row: 'Key' (empty) and 'Value' (empty). Below the table, there are tabs for 'Body', 'Cookies (1)', 'Headers (10)', and 'Test Results'. The 'Body' tab is selected and displays a JSON response with line numbers. The response is:

```

1
2   "success": true,
3   "data": [
4     {
5       "id": 17,
6       "notification_type": "personal",
7       "type": "Transactions",
8       "description": "New Transaction of 11 added for customer (Singh General Store)",
9       "created_at": "2079-12-05T18:15:00.000Z",
10      "status": "UNREAD"
11    },
12    {
13      "id": 18,
14      "notification_type": "personal",
15      "type": "Transactions",
16      "description": "New Transaction of 222 added for customer (Singh General Store)",
17      "created_at": "2079-12-05T18:15:00.000Z",
18      "status": "UNREAD"
19    },
20    {
21      "id": 19,
22      "notification_type": "personal",
23      "type": "Transactions",
24      "description": "New Transaction of 122 added for customer (H Tigers Footware)",
25      "created_at": "2079-12-05T18:15:00.000Z"
}

```

Figure 203:Screenshot of successful response for user notification

4.1.19. Test 19: Checking whether Invoice generated are being sent to the customer email.

Test	19
Description	Checking whether invoice generated are being sent to the customer email.
Action	Creating a new invoice for a customer and checking for sent email.
Expected Result	Email with attachment should be sent to client email.
Actual Result	Email with attachment was sent to client email.
Conclusion	Success

Table 38: Table for testing whether customer receive email.

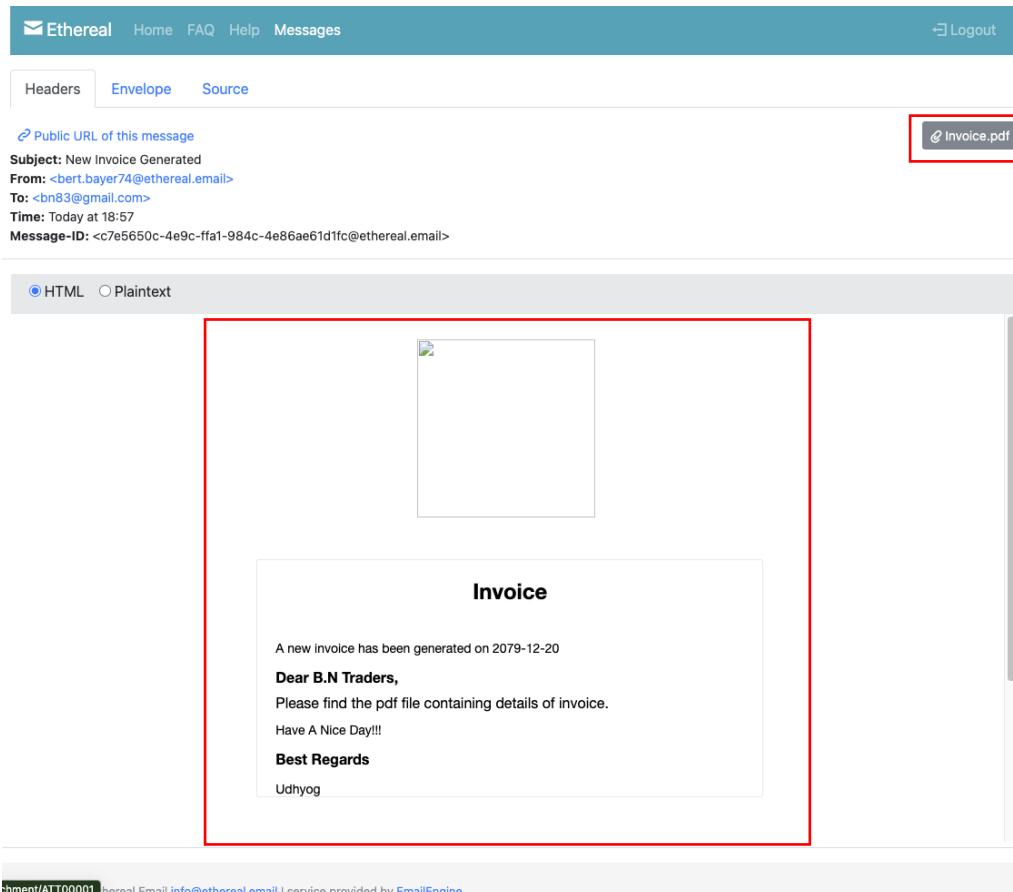
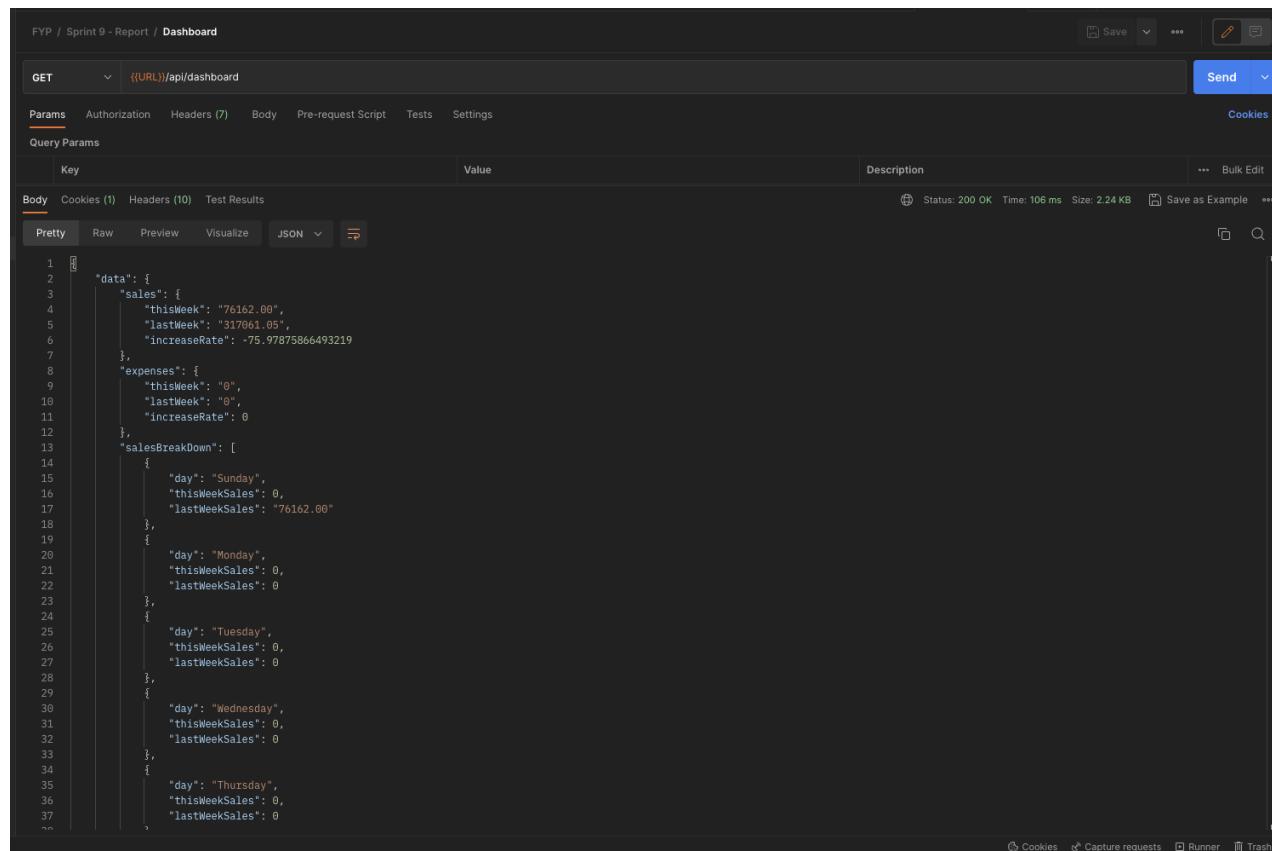


Figure 204: Screenshot of sent email preview where customer received the invoice.

4.1.20. Test 20 - API testing for getting supervisor dashboard report.

Test	20
Description	Testing for getting supervisor data for the dashboard.
Action	Sending API request for getting supervisor dashboard data.
Expected Result	A success response should be returned with dashboard data.
Actual Result	A success response was returned with dashboard data.
Conclusion	Success

Table 39: Testing to get supervisor dashboard data



The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** {{URL}}/api/dashboard
- Params:** (selected tab)
- Headers:** (10 items listed)
- Body:** (empty)
- Query Params:** (empty)
- Tests:** (empty)
- Settings:** (empty)
- Status:** 200 OK (Time: 106 ms, Size: 2.24 KB)
- Actions:** Save as Example, Bulk Edit
- Pretty:** (selected tab)
- Raw:**
- Preview:**
- Visualize:**
- JSON:** (dropdown menu)

```

1  "data": {
2    "sales": {
3      "thisWeek": "76162.00",
4      "lastWeek": "317061.05",
5      "increaseRate": -75.97875866493219
6    },
7    "expenses": {
8      "thisWeek": "0",
9      "lastWeek": "0",
10     "increaseRate": 0
11   },
12   "salesBreakDown": [
13     {
14       "day": "Sunday",
15       "thisWeekSales": 0,
16       "lastWeekSales": "76162.00"
17     },
18     {
19       "day": "Monday",
20       "thisWeekSales": 0,
21       "lastWeekSales": 0
22     },
23     {
24       "day": "Tuesday",
25       "thisWeekSales": 0,
26       "lastWeekSales": 0
27     },
28     {
29       "day": "Wednesday",
30       "thisWeekSales": 0,
31       "lastWeekSales": 0
32     },
33     {
34       "day": "Thursday",
35       "thisWeekSales": 0,
36       "lastWeekSales": 0
37     },
38   ]
39 }
40 
```

Figure 205: Screenshot of API response for dashboard

4.1.21. Test 21 - Testing whether new sales made will be received in the dashboard API.

Test	21
Description	Testing whether new sales made will be received in the dashboard API.
Action	Adding new sales record on Sunday to check whether it is recorded correctly.
Expected Result	The record should be added successfully and recorded in correct day of the week.
Actual Result	The record was added successfully. However, the day of the sales was not returned in correct format.
Conclusion	Failed

Table 40: Testing to review record of sales.

Result:

```
"salesBreakDown": [
    {
        "day": "Sunday",
        "thisWeekSales": 0,
        "lastWeekSales": "76162.00"
    },
    {
        "day": "Monday",
        "thisWeekSales": "41810.00",
        "lastWeekSales": 0
    },
    {
        "day": "Tuesday",
        "thisWeekSales": 0,
        "lastWeekSales": "10000.00"
    },
    {
        "day": "Wednesday",
        "thisWeekSales": 0,
        "lastWeekSales": "10000.00"
    },
    {
        "day": "Thursday",
        "thisWeekSales": 0,
        "lastWeekSales": "10000.00"
    },
    {
        "day": "Friday",
        "thisWeekSales": 0,
        "lastWeekSales": "10000.00"
    },
    {
        "day": "Saturday",
        "thisWeekSales": 0,
        "lastWeekSales": "10000.00"
    }
]
```

Figure 206: Screenshot of incorrectly formatted API response

4.1.22. Test 22 - Testing API for generating reports.

Test	22
Description	Testing API for generating reports.
Action	<p>Sending API request with below query for generating report.</p> <p>startDate=2079-11-02</p> <p>endDate=2079-11-20</p> <p>reportBy=profitbycustomer</p>
Expected Result	The data for the report should be returned in the response.
Actual Result	The data for the report was returned in the response.
Conclusion	Success

Table 41: Testing for generating report by supervisor.

```

FYP / Sprint 9 - Report / Report
POST http://localhost:5100/api/report?startDate=2079-11-02&endDate=2079-11-20&reportBy=profitbycustomer
Params Authorization Headers (8) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1
Body Cookies (1) Headers (10) Test Results
Pretty Raw Preview Visualize JSON
1
2   "message": "Success",
3   "success": true,
4   "data": [
5     {
6       "id": 40,
7       "name": "Singh General Store",
8       "total_sales": "500974.20",
9       "total_cost_price": "313800"
10      },
11      {
12        "id": 41,
13        "name": "B.N Traders",
14        "total_sales": "112322.00",
15        "total_cost_price": "69700"
16      },
17      {
18        "id": 42,
19        "name": "Lumbini Fancy Store",
20        "total_sales": "0",
21        "total_cost_price": "0"
22      },
23      {
24        "id": 43,
25        "name": "W Tidore Foothware"
}

```

Figure 207: Screenshot of API response for report requested.

4.1.23. Test 23 - Testing for downloading data in PDF format.

Test	23
Description	Testing for downloading data in PDF format.
Action	Downloading the details for customer.
Expected Result	A pdf file should be downloaded named as customer.pdf
Actual Result	A pdf file was downloaded named as customer.pdf. However, the format for the pdf was not maintained.
Conclusion	Failed

Table 42: Testing for downloading in PDF format.

The screenshot shows a 'Customers' page with a table of data. The table columns are: ID, Name, Location, Phone, District, Balance, Status, and Actions. The status column includes checkboxes for 'ID', 'Name', 'Location', and 'Phone'. The actions column includes 'Active', 'Download PDF' (which is highlighted with a red box), 'Download CSV', 'Upload CSV', and other options like 'Not Active' and 'Delete'. The customer data is as follows:

ID	Name	Location	Phone	District	Balance	Status	Actions
43	H Tigers Footware	New Road, Kathmandu	98189878432	Baitadi	-690242.90	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active
92	AB Store	Birjung, Nepal	98761222122	Chitawan	0.00	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active
44	AB Store	Birjung, Nepal	98761222122	Chitawan	-24521.00	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active
41	B.N Traders	New Road, Kathmandu	015329953	Bhaktapur	-367063.55	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Not Active
93	Singh General Store	Dhangadhi	98232133133	Kailali	0.00	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active
40	Singh General Store	Dhangadhi	98232133133	Kailali	-665050.20	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active
42	Lumbini Fancy Store	Lumbini, Nepal	9821612782	Syangja	-107937.60	<input checked="" type="checkbox"/> ID <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Location <input checked="" type="checkbox"/> Phone	Active

Page 1 of 1 | Go to page: 1 Total: 7

Figure 208: Screenshot of downloading in PDF format.

ID	Name	Location	Phone	District
43	HTigersFootware	NewRoad,Kathmandu	98189878432	Baitadi
92	AB Store	Birjung,Nepal	98761222122	Chitawan
44	AB Store	Birjung,Nepal	98761222122	Chitawan
41	BN Traders	NewRoad,Kathmandu	015329953	Bhaktapur

Figure 209: Screenshot of wrongly added customers.

A detailed explanation on the solution of the problem has been explained in problem file.

ID	Name	Location	Phone	District	Balance	Status	Action
43	HTigersFootware	New Road, Kathmandu	98189878432	Baitadi	-690342.90	● Active	⋮
92	AB Store	Birjung, Nepal	98761222122	Chitawan	0.00	● Active	⋮
44	AB Store	Birjung, Nepal	98761222122	Chitawan	-24521.00	● Active	⋮
41	BN Traders	New Road, Kathmandu	015329953	Bhaktapur	-367063.55	● Not Active	⋮
93	Singh General Store	Dhangadhi	98232133133	Kailali	0.00	● Active	⋮
40	Singh General Store	Dhangadhi	98232133133	Kailali	-665050.20	● Active	⋮
42	Lumbini Fancy Store	Lumbini, Nepal	9821612782	Syangja	-107937.60	● Active	⋮

Figure 210: Screenshot of downloading customer in correct format.

The process to solve the problem is discussed in artefacts.

4.1.24. Test 24: Testing for downloading data in CSV format.

Test	24
Description	Testing for downloading data in CSV format.
Action	Downloading the details for employees.
Expected Result	A csv file should be downloaded named as employee.csv
Actual Result	A csv file was downloaded named as employee.csv
Conclusion	Success

Table 43: Testing to downloading in CSV format.

The screenshot shows a web-based application interface for managing customers. On the left, there's a sidebar with various menu items: Dashboard, Customers (which is selected and highlighted in blue), Products, Productions, Invoices, Transactions, Employees, and Expenses. The main content area is titled 'Customers' and shows a table of customer data. The table has columns for ID, Name, Location, Phone, District, Balance, and Status. At the bottom right of the table, there are several filter checkboxes (ID, Name, Location, Phone) and two buttons: 'Download PDF' and 'Download CSV'. The 'Download CSV' button is highlighted with a red rectangle. Below the table, there are navigation links for 'Page 1 of 1', 'Go to page: 1', 'Total: 7', and some pagination arrows. At the very bottom, there are buttons for 'Previous' and 'Next'.

Figure 211: Screenshot of downloading in CSV format.

Table data was imported successfully

Name	Location	Phone	Balance	District	PAN	Email	Status
H Tigers Footware	New Road, Kathmandu	98189878432	-690242.90	1	395417549	info@htigers.com	1
AB Store	Birjung, Nepal	98761222122	0.00	5	417925542	ab@gmail.com	1
AB Store	Birjung, Nepal	98761222122	-24521.00	5	417925549	ab@gmail.com	1
B.N Traders	New Road, Kathmandu	15329953	-367063.55	27	604326669	bn83@gmail.com	0
Singh General Store	Dhangadhi	98232133133	0.00	30	482914212	harka@gmail.com	1
Singh General Store	Dhangadhi	98232133133	-665050.20	30	482914812	harka@gmail.com	1
Lumbini Fancy Store	Lumbini, Nepal	9821612782	-107937.60	72	666555121	info@limbinifancy.com	1

Figure 212: Screenshot of downloaded CSV file

4.1.25. Test 25 - Uploading data from CSV format.

Test	25
Description	Testing for uploading the employee's details from csv file.
Action	Uploading the employee's csv.
Expected Result	All the employee's details should be inserted, and number of data inserted should be given to the user.
Actual Result	All the employee's details were inserted, and number of data inserted was given to the user.
Conclusion	Success

Table 44: Testing for adding data from CSV.

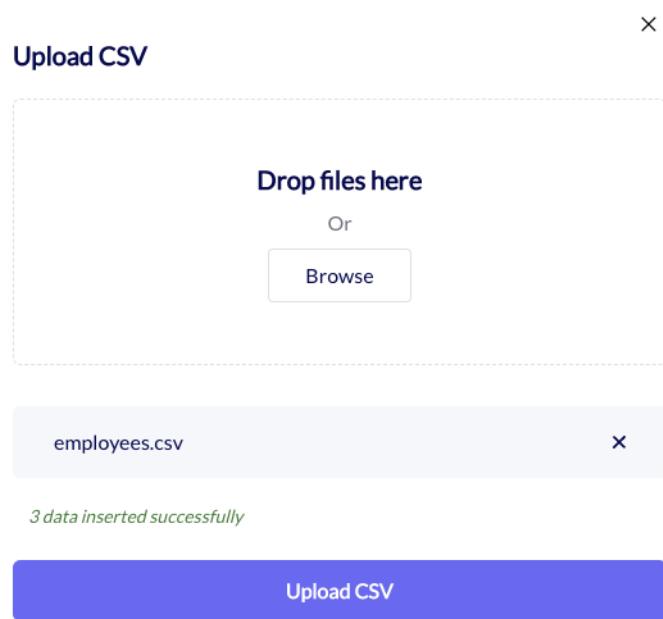


Figure 213: Screenshot of data been successfully added from csv file.

4.1.26. Test 26 - API testing for generating packages with products.

Test	26
Description	API testing for generating packages with products
Action	Passing below details to create new package. date: 2079-12-10 products:[{ "product_id":94, "quantity":200 }, { "product_id":176, "quantity":200 }]
Expected Result	A success message should be received in response.
Actual Result	A success message was received in response.
Conclusion	Success

Table 45: Testing for adding new packages.

Result:

The screenshot shows a Postman interface for a POST request to {{URL}}/api/production. The request body is a JSON object containing a date and a list of products. The response body is a JSON object indicating success and a message.

```
POST {{URL}}/api/production
{
  "date": "2079-12-10",
  "products": [
    {
      "product_id": 94,
      "quantity": 200
    },
    {
      "product_id": 176,
      "quantity": 200
    }
  ]
}

{
  "success": true,
  "message": "Data added successfully"
}
```

Figure 214: Screenshot of success response from API

4.1.27. Test 27 - Testing to generate QR code for production package.

Test	27
Description	Testing to generate QR code for production package.
Action	Requesting to generate QR Code for the package.
Expected Result	A QR Code should be generated for the requested package.
Actual Result	A QR Code was generated for the requested package.
Conclusion	Success

Table 46: Testing to generate QR Code for production package.

ID	Date	Available	Storage Room	Action
DIDF	2079-11-04	Available	R001	⋮
JXN0	2079-11-05	Available	R001	Edit Production View QR Delete Production
Tw9u	2079-11-04	Available	R001	⋮
frsJ	2079-11-07	Available	R002	⋮
jbGi	2079-11-10	Available	R001	⋮
K13h	2079-12-10	Available		⋮

Page 1 of 1 | Go to page: 1 Total: 6

Figure 215: Screenshot for generating QR Code for a package.

Please print the QR Code

You can scan the QR inside Add Invoice to include the package to Invoice

Cancel Print

Figure 216: Screenshot of QR code generated for a package.

4.1.28. Test 28: Testing to add products from package to the invoice.

Test	28
Description	Testing to add products from package to the invoice.
Action	Scanning the QR Code to add the products inside a package.
Expected Result	All the details of the products in package should be displayed and ask for user permission to add.
Actual Result	All the details of the products in package were displayed and ask for user permission to add.
Conclusion	Success

Table 47: Testing for adding products by scanning package.

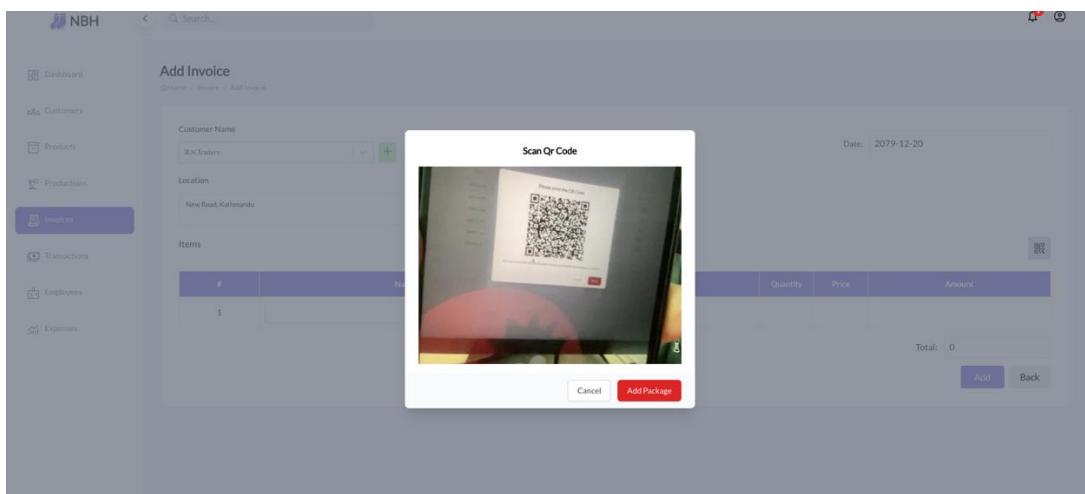


Figure 217: Screenshot of scanning QR.



Figure 218: Screenshot of package details from above scanned QR

4.1.29. Test 29: Testing to add same package multiple times.

Test	29
Description	Testing to add same package multiple times
Action	Scanning the already added QR Code to add the products inside a package.
Expected Result	Error message should be given warning user about the invoice status
Actual Result	Error message was given warning user about the invoice status
Conclusion	Success

Table 48: Testing for adding same package to the invoice.

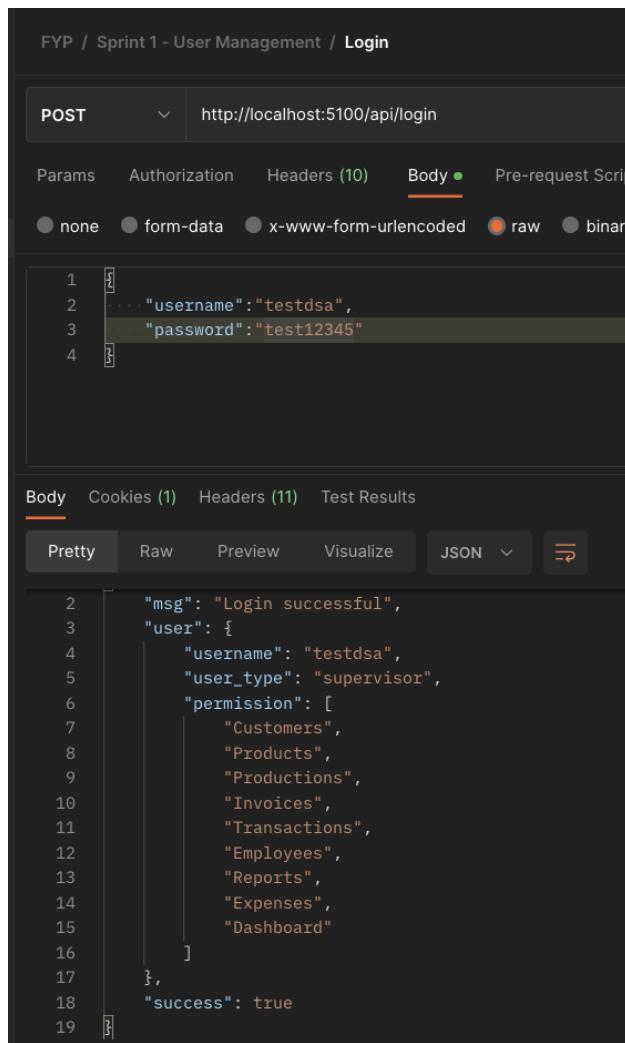
The screenshot shows a software application window titled 'Add Invoice'. At the top left, there's a breadcrumb navigation: Home / Invoice / Add Invoice. On the right side, a red rectangular box highlights an error message: 'Package Already added in this invoice' with a small info icon. Below this, the main form has sections for 'Customer Name' (with a dropdown and a green '+' button), 'Location' (with a text input field), and 'Items' (a table). The 'Items' table has columns: #, Name, Item code, Quantity, Price, and Amount. It contains three rows of data: 1. Fashion Socks (Item code 94, 60 units at 540 each, totaling 32400), 2. Ladies Ankle Sock (Item code 95, 120 units at 600 each, totaling 72000), and 3. (empty row with a green '+' button). At the bottom, there are summary fields: Total: 104400, Discount: 0, % = 0, VAT: 13572, Grand Total: 117972, and buttons for 'Add' and 'Back'.

Figure 219: Screenshot of error message being displayed.

4.1.30. Test 30: Testing authorized route to access users API as supervisor.

Test	30
Description	Testing API to access users API as supervisor
Action	Logging in as supervisor and performing actions only allowed for admin of the application
Expected Result	Error response should be given with proper instruction
Actual Result	Error response was given.
Conclusion	Success

Table 49: Testing to access unauthorized API.



The screenshot shows a Postman interface for a 'Login' request. The method is POST, the URL is `http://localhost:5100/api/login`, and the body contains the following JSON:

```

1  {
2     "username": "testdsa",
3     "password": "test12345"
4 }

```

The response body is displayed in Pretty format:

```

2   "msg": "Login successful",
3   "user": {
4       "username": "testdsa",
5       "user_type": "supervisor",
6       "permission": [
7           "Customers",
8           "Products",
9           "Productions",
10          "Invoices",
11          "Transactions",
12          "Employees",
13          "Reports",
14          "Expenses",
15          "Dashboard"
16      ]
17  },
18  "success": true
19

```

Figure 220: Screenshot of logging in as supervisor

The screenshot shows a POST request to `http://localhost:5100/api/register`. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "email": "supervisor@gmail.com",  
3   "password": "tst",  
4   "username": "tst",  
5   "assignedRole": "supervisor",  
6   "confirm_password": "tst"  
7 }
```

The response status is 403 Forbidden, with the message: "success": false, "message": "You dont have permission to perform this action. Unauthorized Action Detected".

Figure 221: Screenshot of error message thrown for accessing unauthorized API

4.1.31. Test 31: Performing same test above but as admin.

Test	31
Description	Testing API to access users API as admin
Action	Logging in as admin and performing actions only allowed for admin of the application
Expected Result	Request task should be completed without any error
Actual Result	Request task was completed without any error

Table 50: Testing authorized API proper access.

```

POST http://localhost:5100/api/login
Body (10)
Params Authorization Headers (10) Body (1)
none form-data x-www-form-urlencoded
1: {
2:   "username": "admin",
3:   "password": "test12345"
4: }
Body Cookies (1) Headers (11) Test Results
Pretty Raw Preview Visualize JSON
1: {
2:   "msg": "Login successful",
3:   "user": {
4:     "username": "admin",
5:     "user_type": "admin",
6:     "permission": [
7:       "User",
8:       "Dashboard"
9:     ]
10:   },
11:   "success": true
12: }

```

Figure 222: Screenshot of logging in as admin

The screenshot shows the Postman application interface for testing an API. The top navigation bar indicates the project is 'FYP / Sprint 1 - User Management / Register'. The request method is set to 'POST' and the URL is 'http://localhost:5100/api/register'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "email": "supervisor@gmail.com",  
3   "password": "tst",  
4   "username": "tst",  
5   "assignedRole": "supervisor",  
6   "confirm_password": "tst"  
7 }
```

Below the body, the 'Test Results' tab is selected, showing the response body in JSON format:

```
1 {  
2   "msg": "You have been registered successfully",  
3   "success": true  
4 }
```

Figure 223: Screenshot of success response for authorized API

4.1.32. Test 32: Testing whether React redirect to correct portal based on user type.

Test	32
Description	Testing whether React redirect to correct portal based on user type.
Action	Logging in as admin and supervisor and staff and waiting to redirect to respective portal.
Expected Result	All the users should be redirected to their respective portal.
Actual Result	All the users were redirected to their respective portal.
Conclusion	Success

Table 51: Testing whether users are directed to respective portal.

The screenshot shows a login interface with the following elements:

- A title "Login" at the top.
- A "Username" field containing "testdsa".
- A "Password" field containing redacted text and a visibility icon.
- A "Remember me" checkbox followed by the text "Remember me".
- A large blue "Login" button at the bottom.

Figure 224: Screenshot of logging in as supervisor

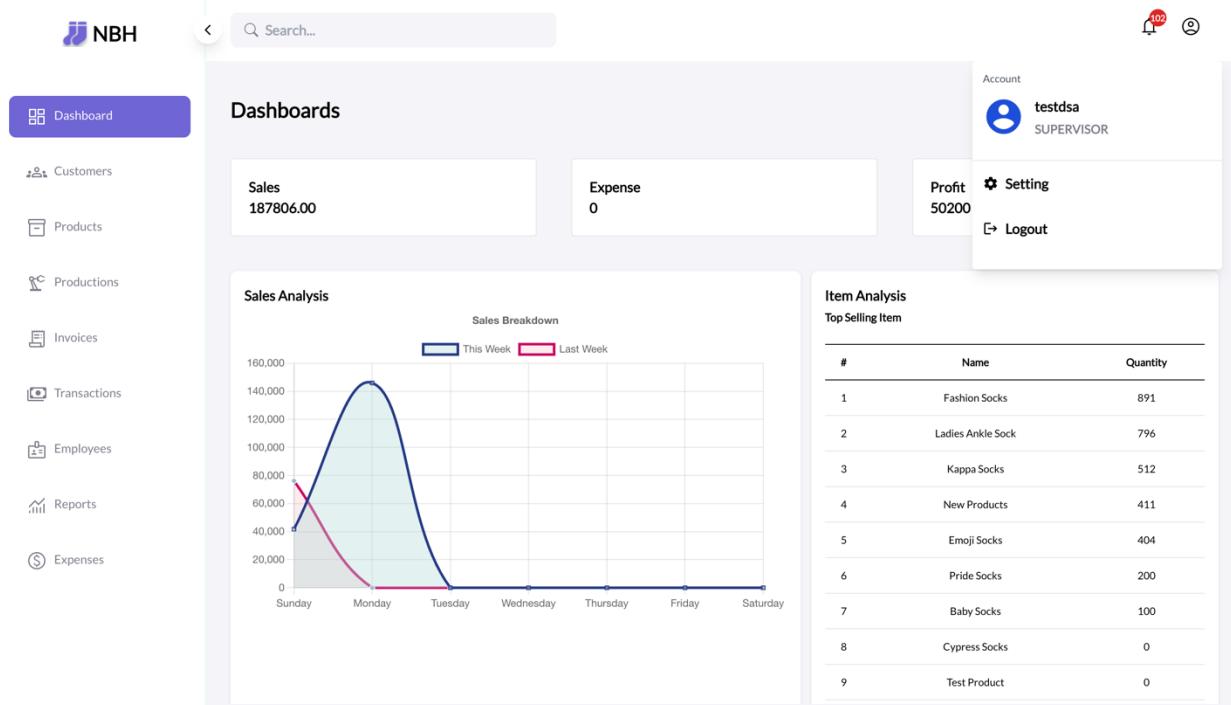


Figure 225: Screenshot of user redirected to supervisor portal.

The screenshot shows a login form titled 'Login'. It has fields for 'Username' (admin) and 'Password' (represented by a series of dots). There is a 'Remember me' checkbox and a large purple 'Login' button at the bottom.

Figure 226: Screenshot of logging in as admin

The screenshot shows a web-based administration interface. At the top left is the logo 'NBH'. A search bar is at the top center. On the right side, there's a user account section for 'admin' (ADMIN) with options for 'Setting' and 'Logout'. Below the header, a purple navigation bar contains icons for 'Dashboard', 'User' (with a count of 8), and other administrative functions. The main content area is titled 'Dashboards' and displays two summary boxes: 'Total Users' (8) and 'Total Supervisor' (6). Below these are four rows of data in a table:

Date	Title	Status
2079-12-24	Employee Attendance	true

At the bottom left, it says 'Page 1 of 1 | Go to page: 1 Total: 4'. At the bottom right are navigation buttons: '<<', 'Previous', 'Next', and '>>'.

Figure 227: Screenshot of user redirected to admin dashboard.

4.1.33. Test 33: Testing to access supervisor portal logged in as staff.

Test	33
Description	Trying to access supervisor page from staff portal
Action	Logging in as supervisor and visiting employees' route from browser URL.
Expected Result	Page not found page should be displayed.
Actual Result	Page not found page was displayed.

Table 52: Testing to access route from direct URL

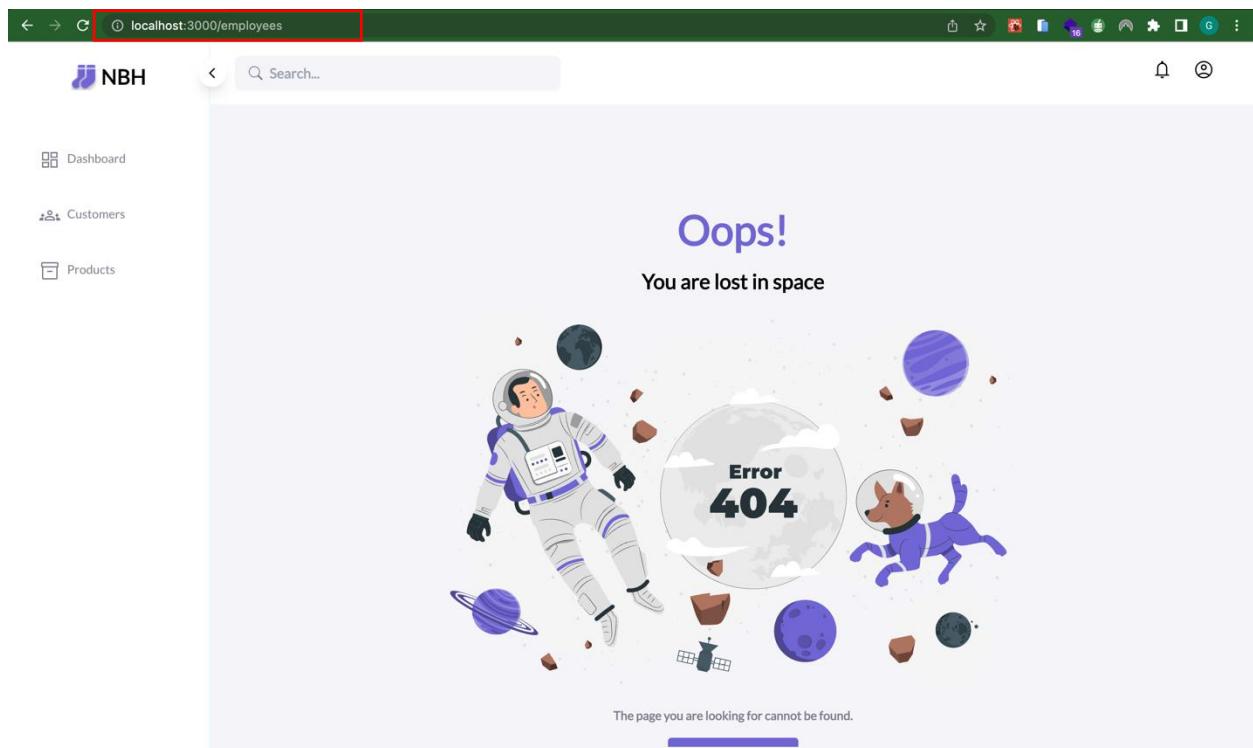
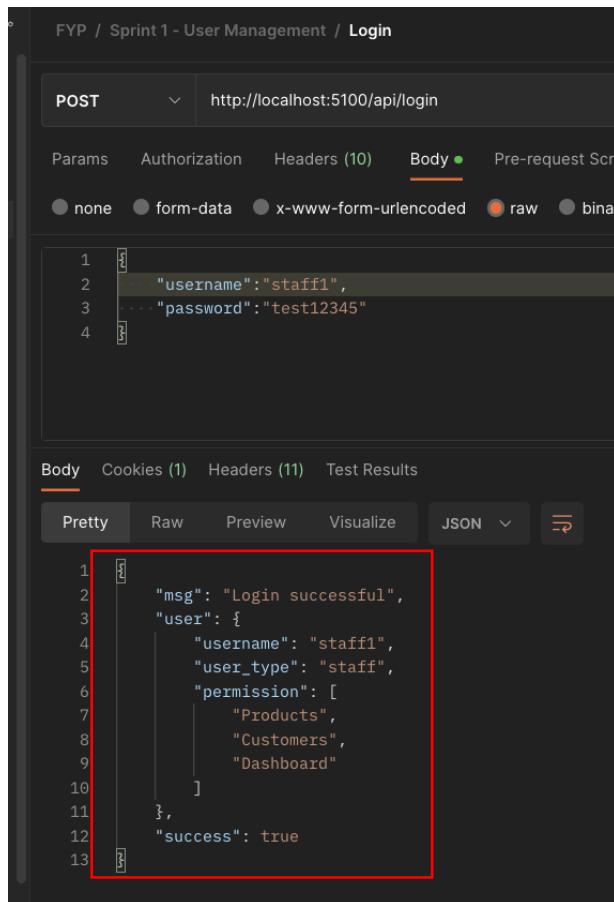


Figure 228: Screenshot of 404 page for accessing unauthorized route.

4.1.34. Test 34: Testing whether staff pages are displayed as per assigned role.

Test	34
Description	Testing whether staff pages are displayed as per assigned role
Action	Logging in as staff and viewing navbar for different options available to the staff member
Expected Result	Proper Navigation Link should be displayed based on user role
Actual Result	Proper Navigation Link was displayed based on user role
Conclusion	Success

Table 53: Testing for viewing custom role for staff.



The screenshot shows a POST request to `http://localhost:5100/api/login`. The Body tab contains the following JSON payload:

```

1 "username": "staff1",
2 "password": "test12345"

```

The response body, shown in Pretty format, is:

```

1 {
2   "msg": "Login successful",
3   "user": {
4     "username": "staff1",
5     "user_type": "staff",
6     "permission": [
7       "Products",
8       "Customers",
9       "Dashboard"
10    ],
11 },
12 "success": true
13 }

```

A red box highlights the permission array in the response body.

Figure 229: Screenshot of staff available permission

The screenshot shows a web-based application interface. At the top left is the logo 'NBH'. A search bar is at the top center. On the far right, there are icons for notifications and user account settings, showing 'staff1 STAFF'. Below the header, a sidebar on the left contains three items: 'Dashboard' (disabled), 'Customers' (selected and highlighted with a red border), and 'Products'. The main content area is titled 'Customers' and shows a table of data. The table has columns: ID, Name, Location, Phone, District, Balance, Status, and Actions. The data in the table is as follows:

ID	Name	Location	Phone	District	Balance	Status	Actions
43	H Tigers Footware	New Road, Kathmandu	98189878432	Balitadi	-690242.90	● Active	⋮
44	AB Store	Birjung, Nepal	98761222122	Chitawan	-24521.00	● Active	⋮
41	B.N Traders	New Road, Kathmandu	015329953	Bhaktapur	-354869.55	● Not Active	⋮
40	Singh General Store	Dhangadhi	98232133133	Kailali	-645050.20	● Active	⋮
42	Lumbini Fancy Store	Lumbini, Nepal	9821612782	Syangja	192062.40	● Active	⋮

At the bottom left, it says 'Page 1 of 1 | Go to page: 1 Total : 5'. At the bottom right are navigation buttons: '<>', 'Previous', 'Next', and '>>'.

Figure 230: Screenshot of available navlink based on user permission.

4.1.35. Test 35: Testing Employee Attendance Cron Job

Test	35
Description	Testing Employee Attendance Cron Job
Action	Changing the cron job time for testing to view the execution of cron
Expected Result	All active employees should be marked as present for today.
Actual Result	All active employees should be marked as present for today.
Conclusion	Success

Table 54: Testing Attendance Cron Job

	280	18	2079-12-25	present
	281	19	2079-12-25	present
	282	20	2079-12-25	present
	283	21	2079-12-25	present
	284	25	2079-12-25	present
	285	8	2079-12-25	present
	286	11	2079-12-25	present
	287	7	2079-12-25	present

Figure 231: Screenshot of data entry for today's attendance

id	date	log	status	description
5	2079-12-25	Employee Attendance	true	

Figure 232: Screenshot of cron log with success

4.1.36. Test 36: Testing Low Stock Alert cron job for supervisors.

Test	36
Description	Testing Low Stock Alert cron job for supervisors.
Action	Changing the cron job time for testing to view the execution of cron
Expected Result	All the supervisors should receive mail with details on low stock
Actual Result	All the supervisor received mail with details on low stock
Conclusion	Success

Table 55: Testing Low stock Corn Job

[🔗 Public URL of this message](#) [Stock.pdf](#)

Subject: Low Stock Alert
From: <bert.bayer74@ethereal.email>
To: <user1@gmail.coim>
Time: Today at 21:16
Message-ID: <a446460f-a5ed-153d-9756-fcb0e7178887@ethereal.email>

HTML Plaintext



Low Stock Alert

Hi User,

Here is a list of low stock items in your inventory. The below list has been generated by an automatic task send by our software.

Have A Nice Day!!

Best Regards

Udhyog

Figure 233: Mail Format for low stock alert

New Bhavuk Hosiery

Kathmandu,Nepal



Low Stock Report

Date: 2023-04-08

Product ID	Product Name	Category	Available Quantity
P005	New Products	Half Socks	89

Figure 234: Screenshot of PDF received in mail.

4.1.37. Test 37: Testing Overdue Alert for all customers.

Test	37
Description	Testing Overdue Alert for all customers.
Action	Changing the cron job time for testing to view the execution of cron
Expected Result	All the customers should receive mail with their overdue invoices
Actual Result	All the customer received mail with their overdue invoices
Conclusion	Success

Table 56: Testing Overdue alert Corn Job

To: <info@htigers.com>	Overdue	Today at 20:54
To: <harka@gmail.com>	Overdue	Today at 20:54
To: <bn83@gmail.com>	Overdue	Today at 20:54
To: <cab@gmail.com>	Overdue	Today at 20:54
To: <info@limbinifancy.com>	Overdue	Today at 20:54

Figure 235: Screenshot for list of email send to all customers.

Subject: Overdue
 From: <bert.bayer74@ethereal.email>
 To: <harka@gmail.com>
 Time: Today at 20:54
 Message-ID: <d5239ab2-d3f8-13bf-a517-f1e24047d495@ethereal.email>

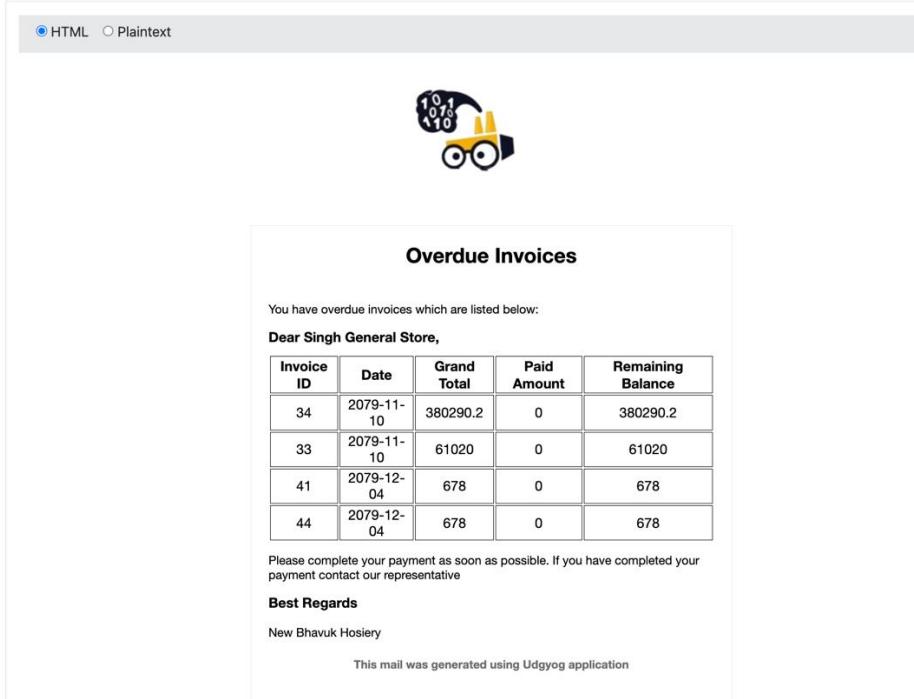


Figure 236: Mail format send to customer.

4.2. Critical Analysis

4.2.1. Analysis on selected methodology

With the completion of the project, choosing scrum taught me development is the least of problem during any software development life cycle. Also, with scrum methodology, I was able to understand different possible changes that might occurs during the development and how can we tackle them. Some of details I learned using SCRUM are as below:

- To successfully complete the assignment, effective time management and self-discipline are essential.
- To complete the job within the allotted time, planning and work prioritization are essential.
- By using an iterative development strategy, we can swiftly adjust to changes and make necessary enhancements.
- The development process can benefit from regular gatherings, like sprint retrospectives, to evaluate your work and pinpoint areas that want development.

4.2.2. Analyzing failed test cases

4.2.2.1. Testing SQL Injection

As the input field was terminating the string, the above error was displayed. Instead of passing values from the request body directly, the data were passed into sql query using template literal placeholder.

```
const username = req.body.username;
const password = req.body.password;
// Checking username in db
const loginQuery = `SELECT * FROM users WHERE username = $1`;
console.log(loginQuery)
db.query(loginQuery, [username], async (err, result) => {
  if (err) {
```

Figure 237: Screenshot of login code after making changes.

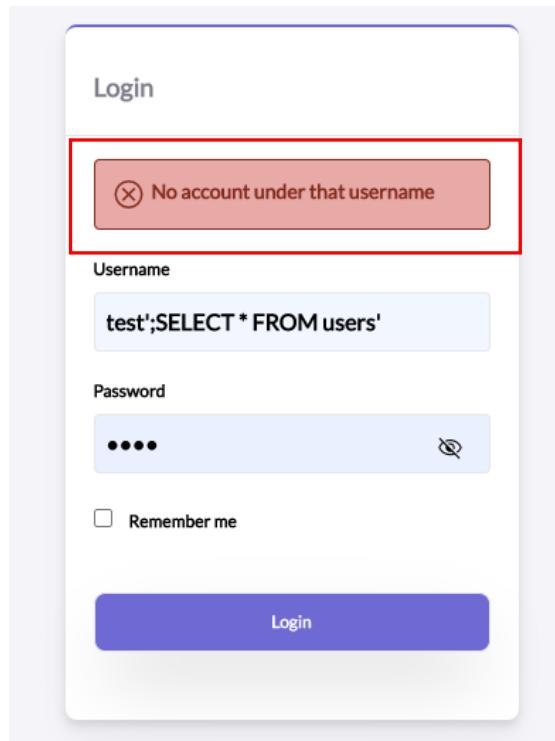


Figure 238: Screenshot of actual result after changes in code

4.2.2.2. Sales breakdown by day testing

As the record were added in BS, the date conversion used wasn't calculating the time difference a causing it to record the details of the sales on wrong day.

```
const lastDayOfThisWeek = adToBs(format(lastDayOfWeek(currentDate), 'yyyy-MM-dd'));
const lastDayofLastWeek = adToBs(format(subDays(lastDayOfWeek(currentDate), 7), 'yyyy-MM-dd'));
const lastDayofPreviousWeek = adToBs(format(subDays(lastDayOfWeek(currentDate), 14), 'yyyy-MM-dd'));
```

Figure 239: Code fix to solve the date format problem.

As the package I used to convert date from ad to bs required the date to be on specific format, converting them using another package “date-fns” solve the issue.

```
{
  "day": "Sunday",
  "thisWeekSales": "41810.00",
  "lastWeekSales": "76162.00"
},
{
  "day": "Monday",
  "thisWeekSales": 0,
  "lastWeekSales": 0
},
```

Figure 240: Screenshot of expected result after code fix

4.2.3. Test Summary

Overall, the developed application performed as expected with passing almost all the test case. All the bugs identified within the application in the period of testing were solved during their respective time frame. All the primary features requested by the client were found to be working as expected.

Likewise, different automatic test cases were also developed and were able to perform as expected. This will help ensure the overall stability of the application for the future. The developed test case will minimize the requirement of manual testing for other future work saving huge amount of time while testing.

Future work should also concentrate on creating a deployable, all-inclusive system for my application.

5. Conclusion

5.1. Legal, Social and Ethical Issues

The section discusses about different issue the project might face or different issue the project might give rise to soon. The section also includes the solution to different issues that might arises.

5.1.1. Legal Issues

5.1.1.1. Unregistered Software

Being an academic project, this project is currently only limited to college. According to law of Nepal, any electronic billing system, billing invoices must be registered to IRD (The Kathmandu Post, 2018). The project not being registered billing system, it cannot be used to generate any of the official invoice bill. Registering the system in IRD seems to be only option for tackle this problem which is discussed in future planning of the project.

5.1.1.2. Collection of Personal Information

The project stores, legal name, address, email, and mobile number of different individuals. According to **Individual Privacy Regulation** of Nepal, only authorized personal of the organization should have access and knowledge of such information (Imperial Law Assosicates, 2021). To upstand such regulation, the application is integrity with authorization and authentication.

5.1.2. Social Issues

5.1.2.1. Privacy Concern

As the system collects and stores personal data about employees, it may raise privacy concerns among workers. The company should be transparent about the data it collects, how it is used, and how it is protected to address these concerns.

5.1.2.2. Intolerance of Change

Some customers could object to the new electronic billing system as they might find use of email to view invoice harder. However, the system allows to print the invoices in which can be viewed in physical format.

5.1.2.3. Workplace exploitation

As the application provide a complete work through for the sales and expense performed inside the industry, this could pressure factory owner on working more harder to boost their profit percentage. The same effect might also be seen in the employees.

5.1.3. Ethical Issues

5.1.3.1. Job Displacement

The implementation of new technology in the factory management system could lead to job displacement for some employees. It's important to consider the impact on employees and to implement measures to minimize negative effects on their livelihoods.

5.1.3.2. Digital Divide

Between personnel who are comfortable with technology and those who are not, there may be a digital divide because of the installation of a new technology-based system. To use the system efficiently, it's crucial to make sure that all staff have access to training and assistance. To solve the issue user interface

5.2. Advantages

5.2.1. Improved Efficiency

A factory management system can help streamline processes and optimize workflows, resulting in improved efficiency and productivity. This can lead to cost savings, increased output, and improved customer satisfaction.

5.2.2. Better Inventory Management

The inventory management feature of a factory management system can help keep track of stock levels, monitor usage rates, and streamline purchasing and expenses. This can help ensure that materials and supplies are always available when needed, reducing production delays and downtime.

5.2.3. Employee Management

The employee management feature of a factory management system can help track employee performance, and their attendance.

5.2.4. Data Analytics and Reporting

A factory management system can provide valuable data analytics and reporting capabilities, allowing managers to track key performance indicators, identify trends and patterns, and make data-driven decisions. This can help improve decision-making and optimize processes.

5.3. Limitations

5.3.1. No Employee Progress Tracking

The system does not have a feature to track individual employee progress or performance. This could limit the ability of managers to identify areas for improvement and optimize employee performance.

5.3.2. Unable to Maintain Batch Wise Production

The system is not designed to track batch-wise production, which could be a limitation for companies that produce products in batches. This could limit the ability of the system to provide accurate inventory management and production planning.

5.3.3. Only Applicable for Single Warehouse

The system is only designed to manage inventory and production processes for a single warehouse. This could be a limitation for companies that operate multiple warehouses, as they may need to implement separate systems for each warehouse.

5.3.4. Manual Employee Attendance

Currently the application is not tied up with any IoT devices causing for supervisor to perform manual attendance for the employee. Adding fingerprint scanner and connecting it with the application would give whole new possibility of tracking shift of different employees, rather than tracking their presence in the industry.

5.4. Future Work

5.4.1. Registering the application with IRD

As discussed in legal issue of this application, currently official tax invoice cannot be generated using this application as it is not registered with IRD. As such registering the developed application to IRD would only make the developed application useful for client.

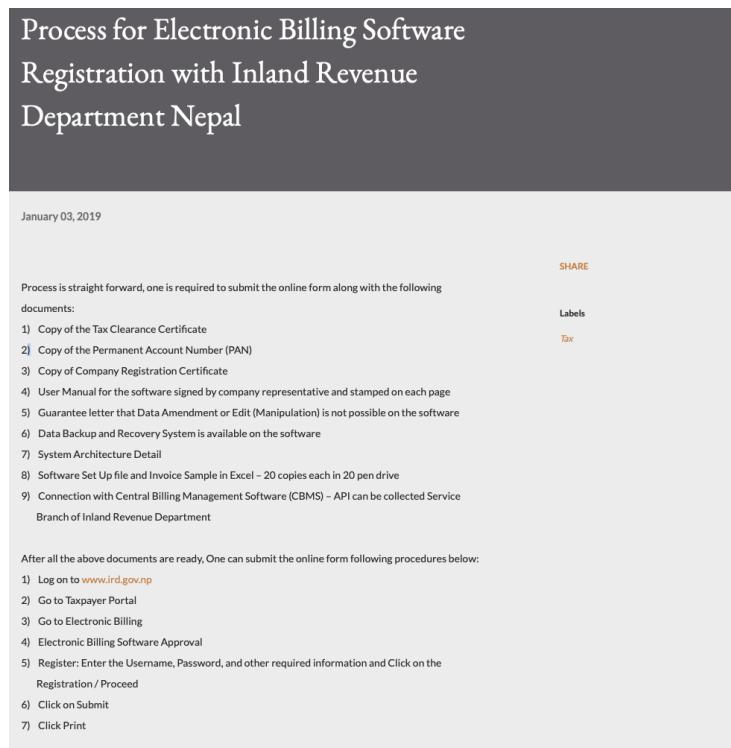


Figure 241: Processes required for registration of CMS.

5.4.2. Batch & Warehouse Wise Production Tracking

The system could be improved to include employee progress tracking for performance monitoring and identifying areas of improvement and updated to track batch-wise production for accurate inventory management and production planning based on batch numbers, production dates, and quality control measures.

5.4.3. Customer Risk Analysis

To increase efficacy and efficiency, the system could benefit from implementing machine learning and AI technologies for predictive maintenance, quality assurance, and demand forecasting. Customer risk calculation could also be added by reviewing customer data,

purchase history, and payment history to avoid risky deals, particularly for credit-based companies.

5.4.4. Mobile Application Development

A mobile app for employees and supervisors could be created to improve accessibility and usability. These could include tools like mobile staff performance tracking, inventory management, and real-time production monitoring.

5.4.5. Integration of the management system with IoT devices.

The system intends to improve the application in the future by connecting IoT devices with the application. This will enable real-time monitoring and management of numerous production operations, resulting in increased efficiency and output. The application can gather and analyze data using IoT devices, enabling predictive maintenance and lowering downtime. This will eventually lead to a more simplified and improved factory management system.

5.4.6. Employee Performance Monitoring

A useful technique for increasing productivity and raising staff engagement is an employee monitoring system. Employers can discover areas for improvement and take remedial action by recording and analyzing employee activity. Furthermore, the system may be used to recognize and reward high-performing staff, which can boost their motivation and job satisfaction. This increased transparency may also result in improved collaboration and a more favorable work atmosphere.

5.4.7. Change the product from CPaaS to SaaS

Currently the entire application is only usable by single client. However, with the vision of this report, I want the application to be in **Software as a Service** where any factory can register to the developed software and implement the software to their industry.

6. References

- actiTIME, 2022. *Waterfall Model: What Is It, When and How to Use It?*. [Online] Available at: <https://www.actitime.com/project-management/what-is-waterfall-model> [Accessed 22 March 2023].
- digite, 2022. *Feature Driven Development: Evolution vs Revolution*. [Online] Available at: <https://www.digite.com/agile/feature-driven-development-fdd/> [Accessed 29 October 2022].
- Geeks for Geeks, 2022. *Software Engineering | Prototyping Model*. [Online] Available at: <https://www.geeksforgeeks.org/software-engineering-prototyping-model/> [Accessed 7 October 2022].
- Henderson, L., 2022. *How To Manage A Sprint Cycle More Effectively*. [Online] Available at: <https://niftypm.com/blog/manage-sprint-cycle/> [Accessed 04 November 2022].
- HiTech Solution & Services Pvt Ltd, 2022. *Swastik Business Accounting Software*. [Online] Available at: <http://hitechnepal.com.np/swastik-business-accounting-software.php> [Accessed 20 December 2022].
- Imperial Law Assosicates, 2021. *Data Protection and Privacy Legislation in Nepal*. [Online] Available at: <https://www.lawimperial.com/data-protection-and-privacy-legislation-in-nepal/> [Accessed 02 April 2023].
- Indeed Editorial Team, 2021. *5 Scrum Phases for Project Management*. [Online] Available at: <https://www.indeed.com/career-advice/career-development/scrum-phases> [Accessed 04 November 2022].
- Ku, L., 2021. *The Impact of Big Data in Business*. [Online] Available at: <https://www.plugandplaytechcenter.com/resources/impact-big-data->

business/

[Accessed 02 November 2022].

- Meta Platforms, 2022. *React.* [Online]
Available at: <https://reactjs.org>
[Accessed 07 November 2022].
- Mountain Goat Software, 2022. *Scrum.* [Online]
Available at: <https://www.mountaingoatsoftware.com/agile/scrum>
[Accessed 23 Spetmeber 2022].
- Nepal Rastriya Bank, 2023. *NEPAL RASTRA BANK.* [Online]
Available at: <https://www.nrb.org.np>
[Accessed 22 March 2023].
- NodeJS, 2022. *Node JS.* [Online]
Available at: <https://nodejs.org/en/>
[Accessed 7 November 2022].
- PostgreSQL, 2022. *PostgreSQL: The World's Most Advanced Open Source Relational Database.* [Online]
Available at: <https://www.postgresql.org>
[Accessed 10 November 2022].
- Product Plan, 2023. *Feature Driven Development (FDD).* [Online]
Available at: <https://www.productplan.com/glossary/feature-driven-development/>
[Accessed 22 March 2023].
- Sharma, K., 2020. *TOP 12 SOFTWARE DEVELOPMENT METHODOLOGIES.* [Online]
Available at: <https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/>
[Accessed 6 October 2022].
- Sharma, L., 2021. *WaterFall Model.* [Online]
Available at: <https://www.toolsqa.com/software-testing/waterfall-model/>
[Accessed 7 October 2022].

- Shrestha, D. P., 2021. *Use of Technology and Its Management Issues in Nepalese Industries and Businesses* , Kathmandu: Nepalese Journal of Management Research.
- Software Testing Help, 2022. *Spiral Model – What Is SDLC Spiral Model?*. [Online] Available at: <https://www.softwaretestinghelp.com/spiral-model-what-is-sdlc-spiral-model/> [Accessed 29 October 2022].
- Tally, 2022. *How To Use Tally.* [Online] Available at: <https://tallsolutions.com/tally/what-is-tally-erp-9-and-how-to-use-it/> [Accessed 20 December 2022].
- Thakur, D., 2023. *Prototyping Model in Software Engineering*. [Online] Available at: <https://ecomputernotes.com/software-engineering/explain-prototyping-model> [Accessed 22 March 2023].
- The Kathmandu Post, 2018. *Guideline on electronic billing system enforced*. [Online] Available at: <https://kathmandupost.com/money/2018/01/18/guideline-on-electronic-billing-system-enforced> [Accessed 02 April 2023].
- Vaypar App Pvt Ltd, 2022. *About.* [Online] Available at: <https://vyaparapp.in/gst-billing-app> [Accessed 20 December 2022].
- w3schools, 2022. *Node.js Tutorial.* [Online] Available at: <https://www.w3schools.com/nodejs/default.asp> [Accessed 07 November 2022].
- Wan, S., Gao , J. & Li, D., 2016. Exploring the Advantages of Content Management Systems for Managing Engineering Knowledge in Product-service Systems. In: *Procedia CIRP*. Amsterdam: Elsevier B.V., pp. 446-450.
- Zoho Inventory, 2022. *Zoho Inventory.* [Online] Available at: <https://www.zoho.com/us/inventory/> [Accessed 24 December 2022].

7. Appendix

7.1. Appendix A – Pre-Survey (Sample)

[Return to top](#)

Survey Form	Project Title: Udyog
Factory Name: Bhattarai Kopada Udyog	
Factory Major Product: Tshirt Crop Top, Shocks	
Contact Details: 9814986843	
<p>1. Does your industry usage any type of software to manage your industry? (के तपाईंको उद्योगले तपाईंको उद्योग व्यवस्थापन गर्नु कुनै प्रकारको सफ्टवेयर प्रयोग गर्दछ?)</p> <p>a) Yes (हामी गर्दौं) <input checked="" type="checkbox"/></p> <p>b) No (हामी गर्दैनौं) <input type="checkbox"/></p>	
<p>2. If your industry usage any software, could you mention the software? (यदि तपाईंको उद्योगले कुनै सफ्टवेयर प्रयोग गर्दछ भने, तपाईं सफ्टवेयर उल्लेख गर्ने सक्नुहुन्छ?)</p> <p>a) Yes (हामी सक्छौं) <input checked="" type="checkbox"/> Tally</p> <p>b) No (हामी सक्दैनौं) <input type="checkbox"/></p>	
<p>3. If you are using a software, are you happy with it? (यदि तपाईं सफ्टवेयर प्रयोग गर्दै हुनुहुन्छ भने, तपाईं त्यसमा खुसी हुनुहुन्छ?)</p> <p>a) I am (खुशी छु) <input type="checkbox"/></p> <p>b) I am not (म खुशी छैन) <input checked="" type="checkbox"/></p>	

Figure 242: Survey Form Sample (2)

Survey Form

Project Title: **Udhyog**

Survey Form

Project Title: Udhyog

4. If your industry does not usage any software, do you know the benefits of moving to digitalization? (यदि तपाईंको उद्योगले कुनै सफ्टवेयर प्रयोग गर्दैन भने, के तपाईंलाई डिजिटलाइजेसनमा सर्व फाइदाहरू थाहा छ?)

a) Yes (हाँ)
b) No (ैन)

5. Have you faced any losses due to human errors in calculation? (के तपाईंले गणनामा मानवीय त्रुटिहरूको कारणले कुनै हानिको सामना गर्नुभएको छ?)

a) Yes (हाँ)
b) No (ैन)
c) Not Sure (थाहा भएन)

6. How do you estimate exact assets, or stock in your company? (तपाईं तपाईंको कम्पनी मा सही सम्पत्ति, वा स्टक कसरी अनुमान गर्नुहुन्छ?)

a) Manually (म्यानुअल रूपमा)
b) Not sure (थाहा भएन)
c) We use software (हामी सफ्टवेयर प्रयोग गर्दौं)

Figure 243: Survey Form Sample (2)

Survey Form		Project Title: Udyog
Survey Form		Project Title: Udyog
<p>7. How do you compare your company performance with past production? (विगतको उत्पादनसँग आफ्नो कम्पनीको प्रदर्शनलाई कसरी तुलना गर्नुहुन्छ?)</p> <p><input checked="" type="checkbox"/> a) Manually (म्यानुअल रूपमा) <input type="checkbox"/> b) Not sure (थाहा भएन) <input type="checkbox"/> c) We use software (हामी सफ्टवेयर प्रयोग गर्छौं)</p>		
<p>8. Do you think you need a software to track activities in your industry? (के तपाईंलाई आफ्नो उद्योगमा गतिविधिहरू इयाक गर्न सफ्टवेयरको आवश्यकता छ जस्तो लाग्छ?)</p> <p><input checked="" type="checkbox"/> a) Yes (छ) <input type="checkbox"/> b) No (हैन) <input type="checkbox"/> c) Not Sure (थाहा भएन)</p>		
<p>9. Why have you not switched to any software if you know the benefits of moving to digitalization? (यदि तपाईंलाई डिजिटलाइजेसनमा सर्व फाइदाहरू थाहा छ भने तपाईंले कुनै सफ्टवेयरमा किन स्विच गर्नुभएन?)</p> <p>I have not found any better solution at my price till now</p>		

Figure 244: Survey Form Sample (3)

Survey Form

Project Title: Udyog

10. If a software for management of industry were to be developed, what are some of necessary feature you want to include ? (यदि उद्योगको व्यवस्थापनको लागि सफ्टवेयर विकास गर्ने हो भने, तपाईं कैहि आवश्यक सुविधाहरू समावेश गर्न चाहनुहुन्छ?)

Purchase Book Sales Book
 Salary Information
 Machine Expenses

Date: 2022 / 10 / 13


Signature

Figure 245: Survey Form Sample (4)

7.2. Appendix B – Design

7.2.1. Gantt Chart Development

[Return to top](#)

Due to changes in the product backlog, same changes were also made to the gantt chart. The order of sprint was changed from the planned. The original Gantt Chart is viewed below:

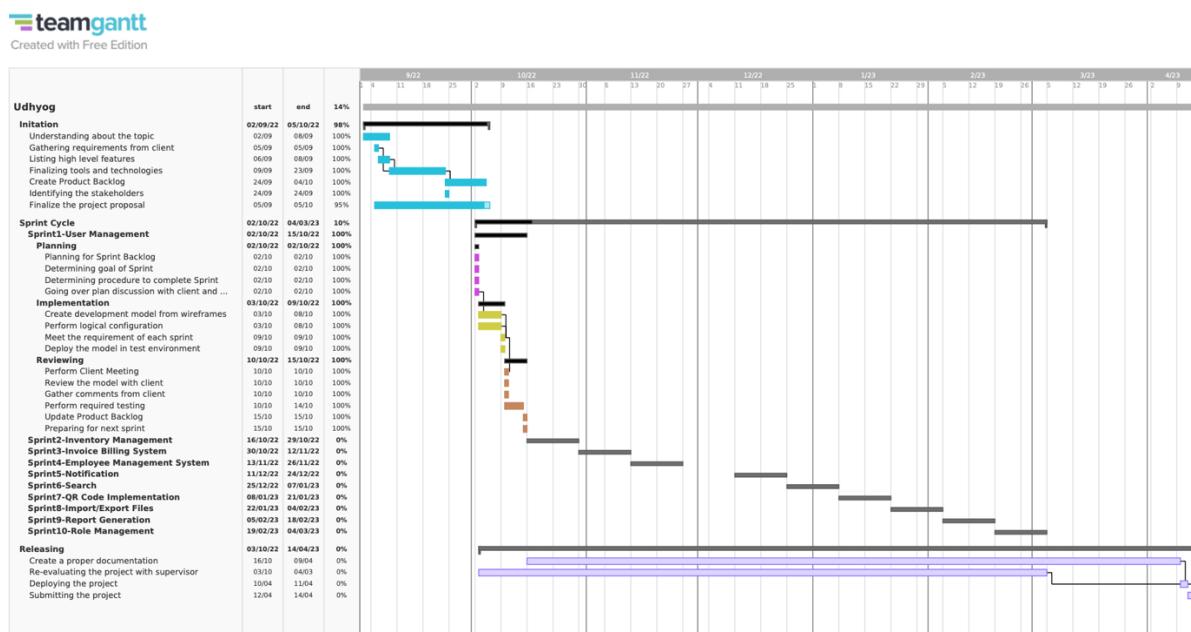


Figure 246: Gantt Chart rehearsal

7.2.2. Use Case Development

[Return to top.](#)

7.2.2.1. First rehearsal of use case diagram

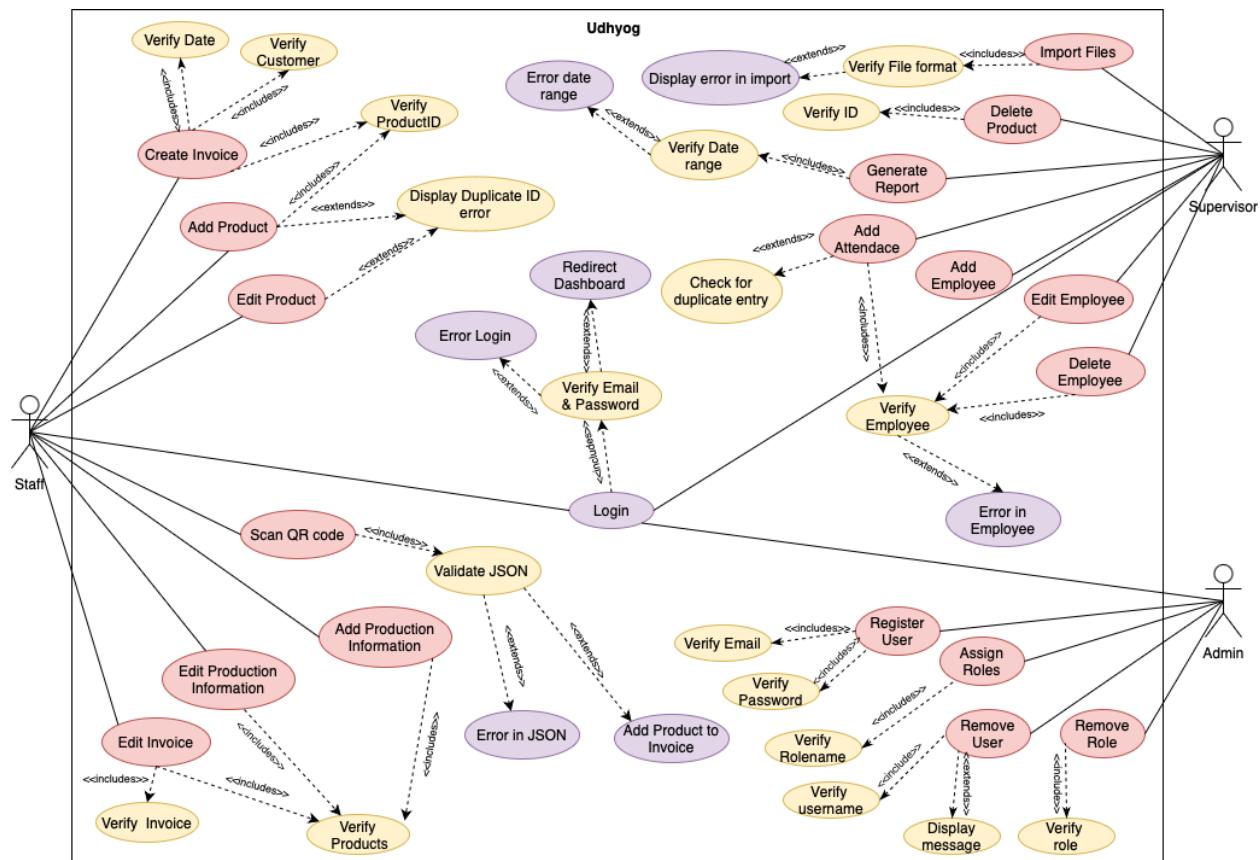


Figure 247: Use Case first rehearsal

7.2.3. ERD Development

[Return to top](#)

7.2.3.1. First Rehearsal of ERD

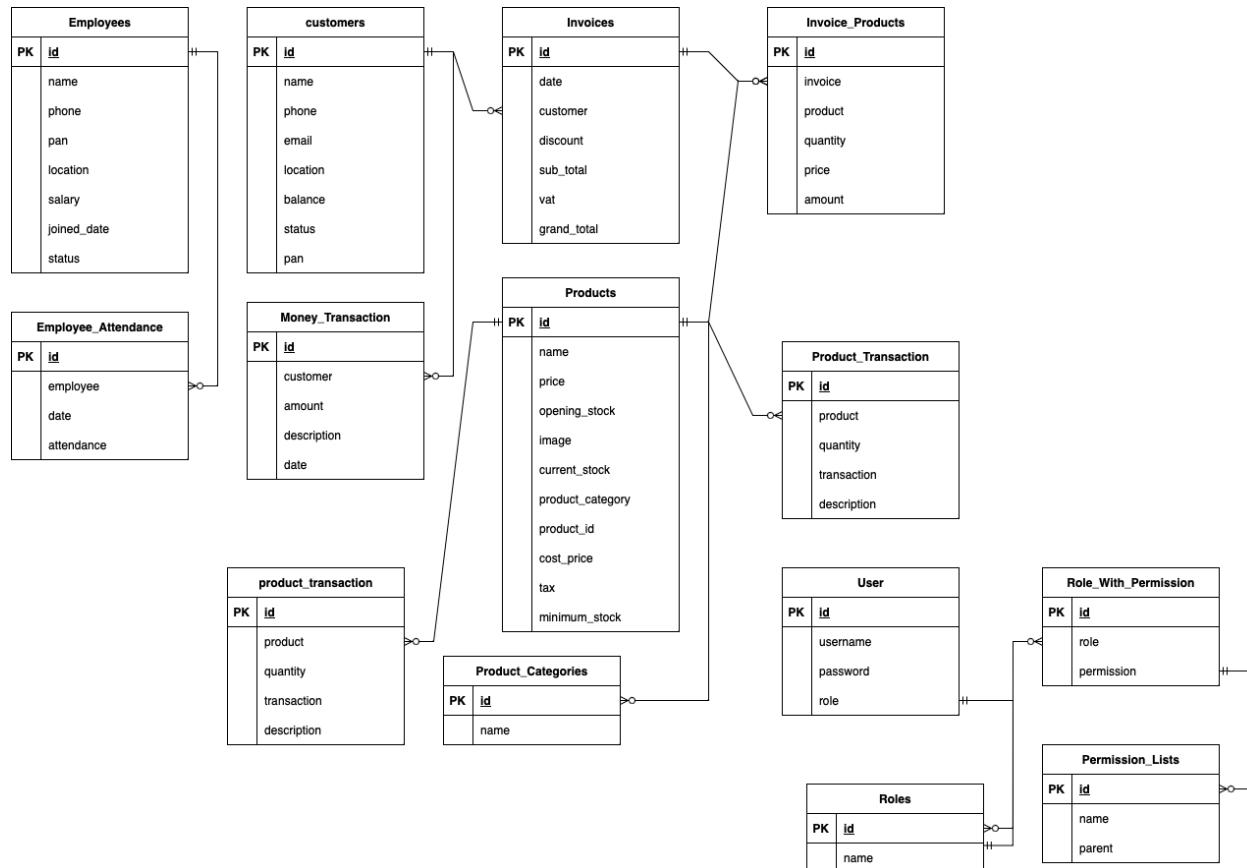


Figure 248: First Iteration of ERD

7.2.3.2. Second Rehearsal of ERD

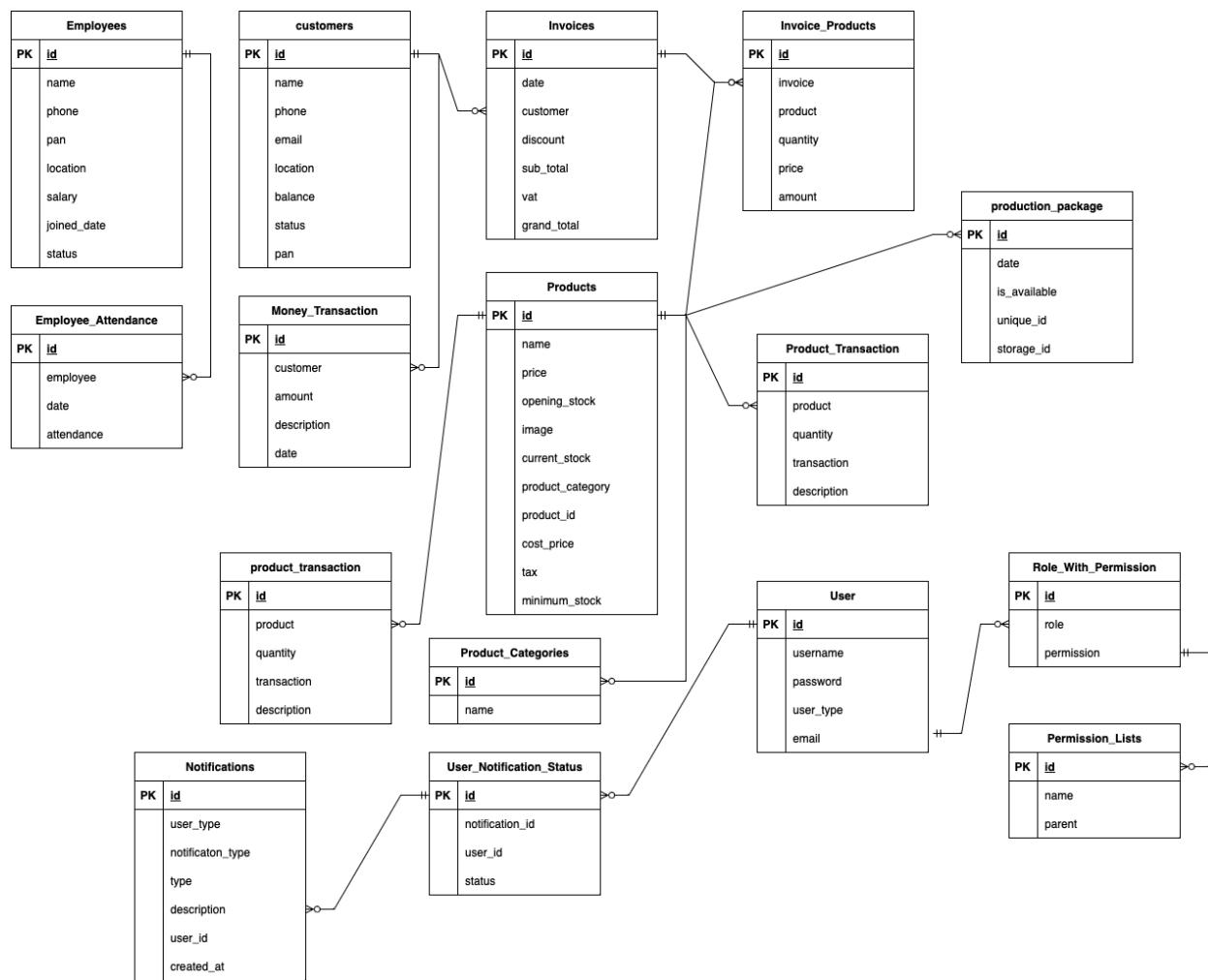


Figure 249: Second Iteration of ERD

7.3. Appendix C - User Feedback

7.3.1. Form Sample

The image shows a sample user feedback form. At the top, it says "User Feedback Form". Below that, a note states: "This form is generated to retrieve user response for the developed system. Please submit the relevant answer for the below questioners." The first section asks for "Your Name *". It has a placeholder "Short-answer text" and a dotted line for input. The second section asks for the user's current employment role. It includes a note: "Please provide your role at any of your current places of employment. *". There are four radio button options: "Staff", "Management Team", "Owner", and "Other...".

User Feedback Form

This form is generated to retrieve user response for the developed system. Please submit the relevant answer for the below questioners.

Your Name *

Short-answer text

Please provide your role at any of your current places of employment. *

Staff

Management Team

Owner

Other...

Figure 250: User Feedback Form (1)

Please rate the overall user experience for the developed application at range of 1-5. 1 being lowest and 5 being highest. *

1
 2
 3
 4
 5

Rate the overall application based on its looks and appearance at range of 1-5. 1 being lowest and 5 being highest. *

1
 2
 3
 4
 5

Figure 251: User Feedback Form (2)

Do you think the application is a useful technology for different industry available in our country?

Yes
 No
 Maybe

How confident do you feel that the application will shift the industry of Nepal to face of digitalization? *

1 2 3 4 5

Not confident at all Yes it can!

Do you think a mobile application for the developed application would be a better approach rather than the web application? *

Yes
 No
 Maybe

Figure 252: User Feedback Form (3)

Do you have any other suggestion that this application can include that might be helpful for industry in Nepal?

Long-answer text

Figure 253: User Feedback Form (4)

7.3.2. Post Survey Result

[Return to Top](#)

Please provide your role at any of your current places of employment.

11 responses

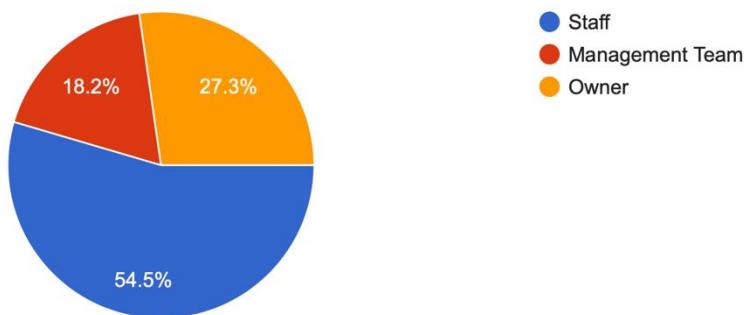


Figure 254: Post survey result (1)

Please rate the overall user experience for the developed application at range of 1-5. 1 being lowest and 5 being highest.

11 responses

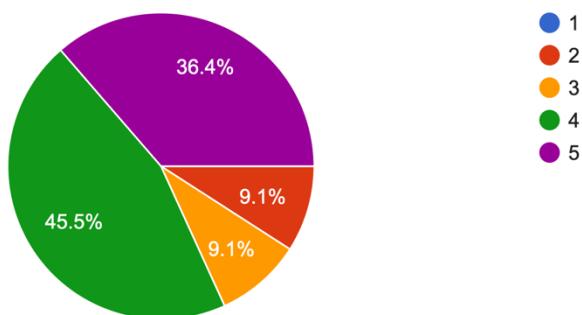


Figure 255: Post survey result (2)

Rate the overall application based on its looks and appearance at range of 1-5. 1 being lowest and 5 being highest.

11 responses

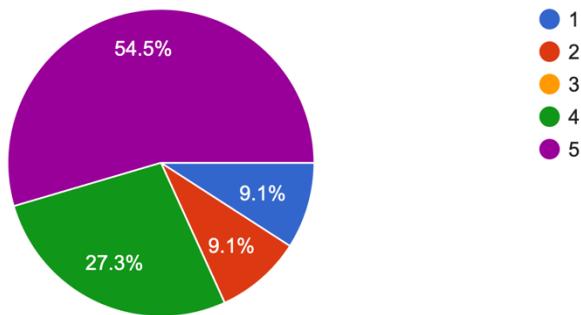


Figure 256: Post survey result (3)

Rate the overall application based on its looks and appearance at range of 1-5. 1 being lowest and 5 being highest.

11 responses

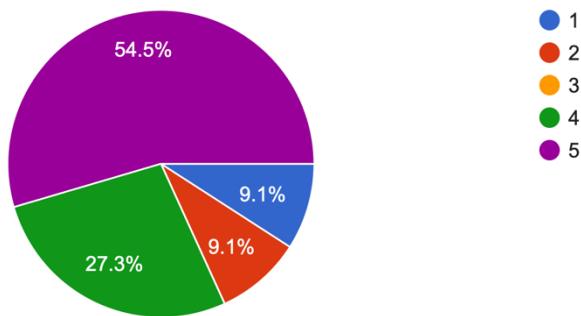


Figure 257: Post survey result (4)

Do you think the application is a useful technology for different industry available in our country?
11 responses

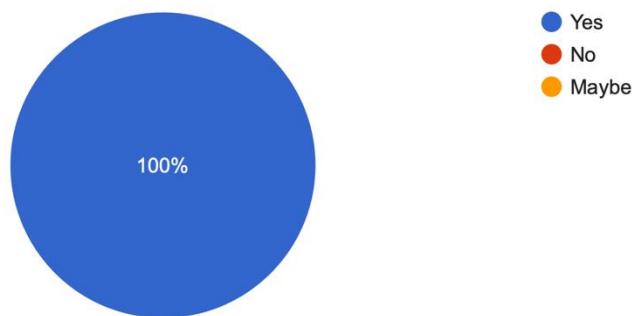


Figure 258: Post survey result (5)

How confident do you feel that the application will shift the industry of Nepal to face of digitalization?

11 responses

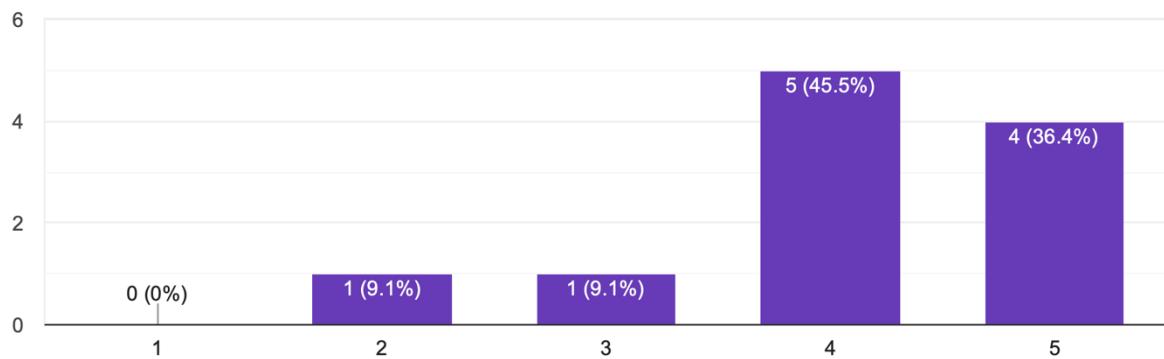


Figure 259: Post survey result (6)

Do you think a mobile application for the developed application would be a better approach rather than the web application?

11 responses

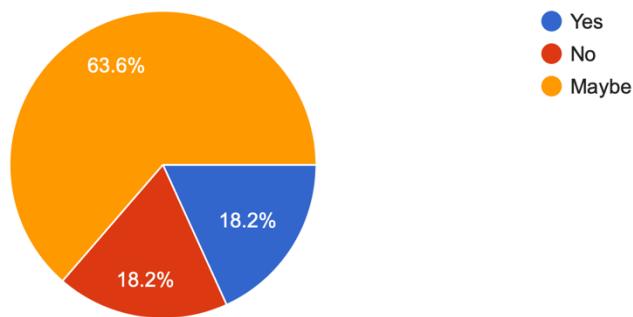


Figure 260: Post survey result (7)

7.3.3. Comparison of Result from post survey and pre survey

A pre-survey was done to determine the requirement for software solutions to track activities before designing the application for tracking activities in various sectors. The poll was given to several specialists from various industries, and 100% of respondents agreed that such software was needed. This aided in the application's development. Following the development and implementation of the application, a post-survey was undertaken to gain feedback on its use. The reaction was overwhelmingly favorable, with 100% of respondents considering the application to be a valuable tool for many businesses in our country. These comments validated the application's potential effect and gave useful insights into areas for future enhancement.

7.4. Appendix D – Similar Projects

[Return to top](#)

7.4.1. Zoho Inventory

Zoho Inventory is a cloud-based application allowing its user to manage the inventory using its features. The application was developed in with aim to solve the problem faced by small and medium size business. The application is available in both web and mobile application. The application has features like customer management, vendor management, order creation and much more (Zoho Inventory, 2022).

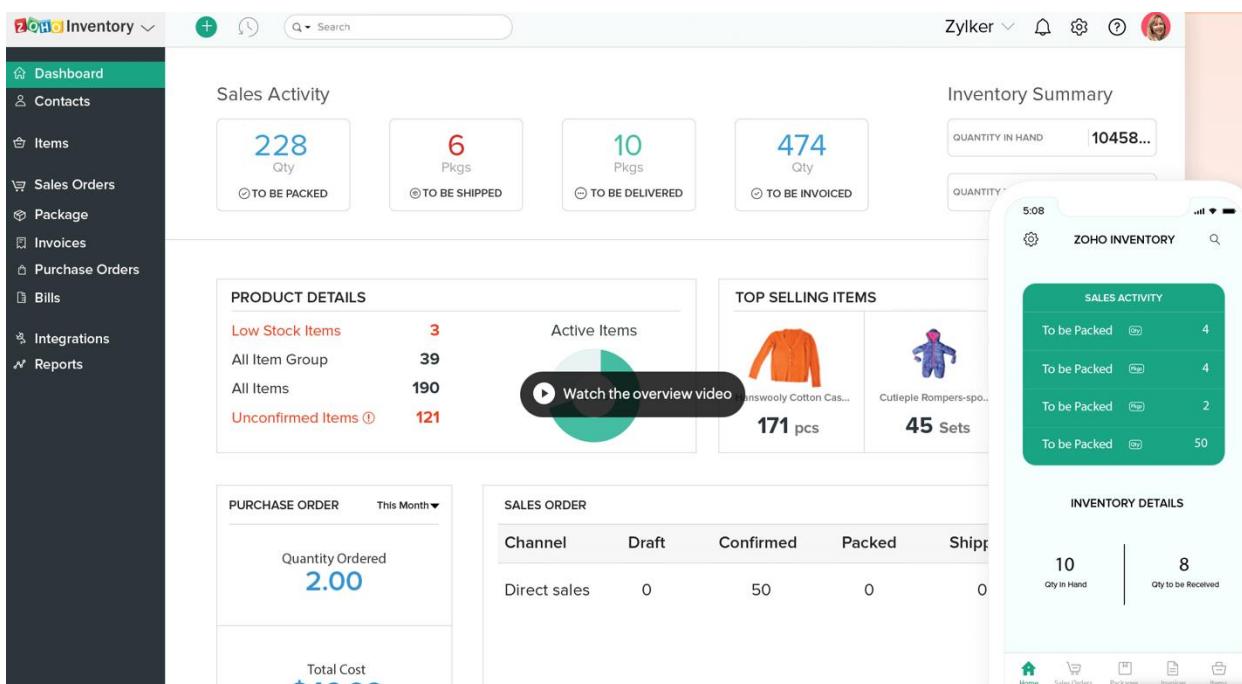


Figure 261: Dashboard of Zoho Inventory

Source: (Zoho Inventory, 2022)

Some of the features included in the Zoho that have inspired this project are:

- Barcode scanning
- Reporting and analytics

7.5. Appendix E – Product Backlog Changes

Due to some unseen event, user notification was added before role management in the product backlog, however without authorization adding notification would make the project revisit the task again, as such it was shifted to Sprint 7 moving Role Based System to Sprint 5 and Search in Sprint 6.

7.5.1. Changes made in Product Backlog

Product Backlog - Udhayog						
ID	As a ..	I want to be able to ..	So that..	Priority	Sprint	Status
PB1	Admin,Staff,Supervisor	login to the system	I can use the system	Must	1	Completed
PB2	Admin	create new user within the system	I can give access to the system	Must	1	Completed
PB4	Staff	maintain my inventory data	I can be aware of my assets	Must	2	Completed
PB5	Supervisor	maintain category of products	content is easily accessible and search	Could	2	Completed
PB7	Staff	create invoice	I can sell my products	Must	3	Completed
PB8	Supervisor	track my employees salary	I can pay them	Must	4	Completed
PB3	Admin	restrict user action in the system	I can create role based system	Should	5	Completed
PB15	Staff, Supervisor	Perform task based on role	I can help admin	Must	5	Completed
PB10	Supervisor	get notification on invoice overdue	I can notify the customer	Should	7	
PB11	Supervisor	get notification on low stocks	I can be aware of my assets	Should	7	
PB12	Staff, Supervisor	search in the system	I can find results	Must	6	Doing
PB6	Staff	maintain information of package	I am able to bill them easily	Should	8	
PB13	Staff	get digital invoice and report	I can share them	Should	9	
PB14	Supervisor	generate report about my business	I can make decision based on them	Must	10	

Figure 262: Changes in order of sprint

7.5.2. Final Product Backlog

Product Backlog - Udhayog						
ID	As a ..	I want to be able to ..	So that..	Priority	Sprint	Status
PB1	Admin,Staff,Supervisor	login to the system	I can use the system	Must	1	Completed
PB2	Admin	create new user within the system	I can give access to the system	Must	1	Completed
PB4	Staff	maintain my inventory data	I can be aware of my assets	Must	2	Completed
PB5	Supervisor	maintain category of products	content is easily accessible and search	Could	2	Completed
PB7	Staff	create invoice	I can sell my products	Must	3	Completed
PB8	Supervisor	track my employees salary	I can pay them	Must	4	Completed
PB3	Admin	restrict user action in the system	I can create role based system	Should	5	Completed
PB15	Staff, Supervisor	Perform task based on role	I can help admin	Must	5	Completed
PB10	Supervisor	get notification on invoice overdue	I can notify the customer	Should	7	Completed
PB11	Supervisor	get notification on low stocks	I can be aware of my assets	Should	7	Completed
PB12	Staff, Supervisor	search in the system	I can find results	Must	6	Completed
PB6	Staff	maintain information of package	I am able to bill them easily	Should	8	Completed
PB13	Staff	get digital invoice and report	I can share them	Should	9	Completed
PB14	Supervisor	generate report about my business	I can make decision based on them	Must	10	Completed

Figure 263: Final Product Backlog

7.6. Appendix F – Considered Methodology

7.6.1. FDD Model

[Return to top](#)

7.6.1.1. Lifecycle/Phases of FDD Model



Figure 264: Lifecycles in FDD Model

Source: (*Product Plan, 2023*)

The FDD model is a simple model having only two phases. i.e.,

- Planning Phase
- Construction Phase

7.6.1.2. Advantages of FDD Model

- The model helps the development team select and finish the most essential features first, assuring on-time release.
- FDD is simply adaptable to big tasks.
- The model emphasizes design to ensure a well-designed, easy-to-maintain product.
- FDD model encourages regular team contact and planning.

7.6.1.3. Roles and Responsibilities

Roles	Responsibilities
Project Manager	<ul style="list-style-type: none"> • Oversees entire project. • Follows traditional methodology of working
Development Manager	<ul style="list-style-type: none"> • Oversees the day-to-day activities of the project. • Mentor of development team member
Chief Architect	<ul style="list-style-type: none"> • Oversees the overall design aspect of the project
Chief Programmer	<ul style="list-style-type: none"> • Oversees the development team with design and analysis
Class Owner	<ul style="list-style-type: none"> • Design, Code, Test

Table 57: Roles and Responsibilities in FDD (digite, 2022)

7.6.2. Spiral Model

[Return to top](#)

7.6.2.1. Lifecycle/Phases of Spiral Model

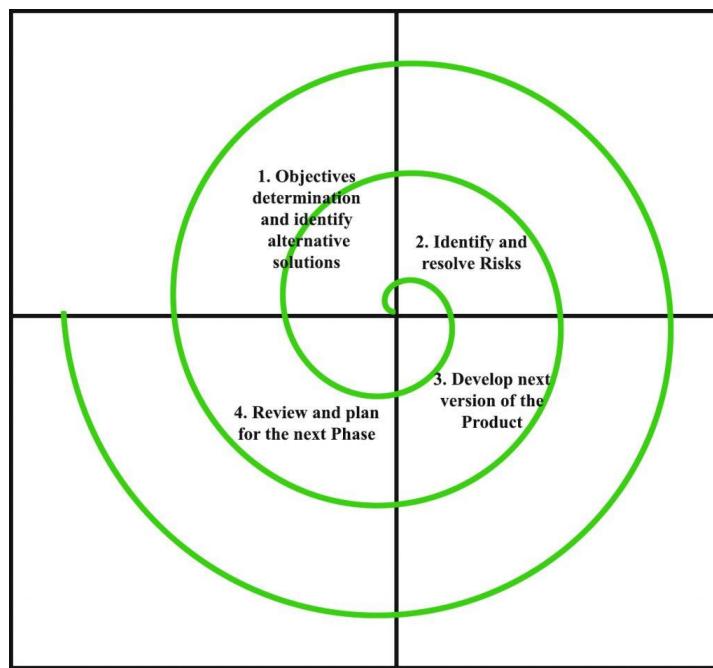


Figure 265: Lifecycles of Spiral Model

Source: (Geeks for Geeks, 2022)

The lifecycle and phases of the spiral model are listed below:

- Planning
- Risk Analysis
- Engineering
- Evaluation

7.6.2.2. Advantages of Spiral Model

- The model lowers project risks by recognizing and managing them early on.
- It allows for alterations and revisions at any level of the development process.
- The Spiral Model provides continual input and review for refinement and development of the final output.

- The model decreases the possibility of mistakes and rework, leading to cost and time efficiency in the development process.

7.6.2.3. Roles and Responsibilities

Roles	Responsibilities
Business Analyst	<ul style="list-style-type: none"> • Gather requirement for the project from the client. • Evaluate risks in the project
Senior Member	<ul style="list-style-type: none"> • Create plans to execute the project. • Evaluate risks in the project
Developers Teams	<ul style="list-style-type: none"> • Creates the small-scale project. • Evaluate risks in the project
Tester	Verify the codes written by developers.

Table 58: Roles and Responsibilities in Spiral Model (Sharma, 2020)

7.6.3. Prototype Model

[Return to top](#)

7.6.3.1. Lifecycle/Phases of Prototype Model

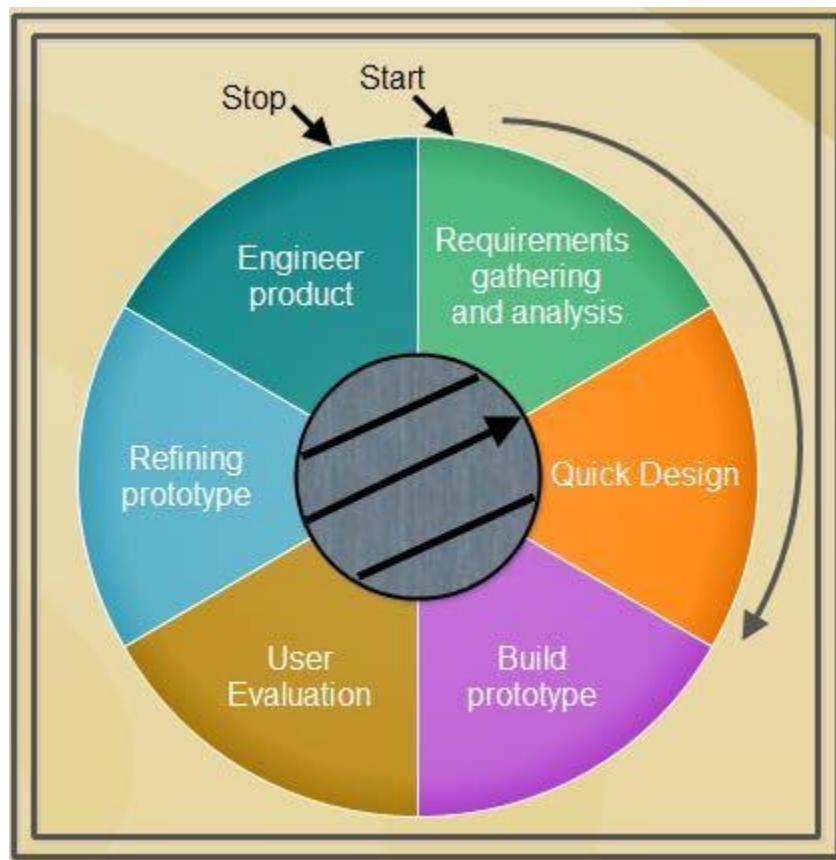


Figure 266: Lifecycles involved in prototype model.

Source: (Thakur, 2023)

Different phases involved in prototype model:

- Requirements gathering
- Quick Design
- Prototype development
- Prototype testing

- Refining the prototype
- Prototype testing
- Final product development
- Testing and deployment

7.6.3.2. Roles and Responsibilities

Roles	Responsibilities
Business Analyst	Gather requirement for the project from the client
Developers Teams	Creates the prototype which are evaluated by the client
Client	Evaluates the prototype built by the developers
Tester	Verify the codes written by developers is ready to use after prototype is accepted.

Table 59: Roles and Responsibilities in Prototype Model (Geeks for Geeks, 2022)

7.6.3.3. Advantages of Prototype Model

- The Prototype model enables developers to obtain fast feedback from stakeholders, decreasing the possibility of expensive rework later.
- The Prototype model allows for experimentation and innovation, resulting to better designs and greater functionality.
- The Prototype model helps to uncover possible errors early in the development process, minimizing the cost and labor necessary to correct them.

7.6.4. Waterfall Model

[Return to top](#)

7.6.4.1. Lifecycle/Phases of Waterfall Model

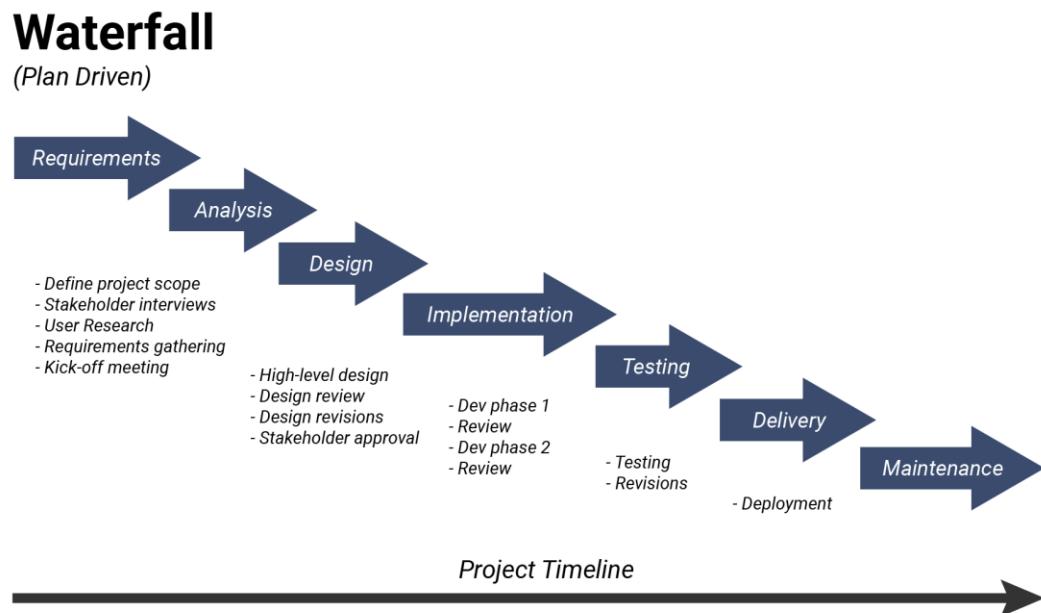


Figure 267: Lifecycles of waterfall model

(Source: (actiTIME, 2022))

As described above, the waterfall model follows SDLC steps in sequential order. Below are the lists of different phases involved in waterfall model:

- Requirement Analysis
- System Design
- Implementation
- Testing
- Deployment
- Maintenance

7.6.4.2. Roles and Responsibilities

While following the waterfall different member of the teams are required to perform different task for completing the project. The team perform their task depending on the phase on which the waterfall model is currently running.

Roles	Responsibilities
Business Analyst	Gather requirement for the project from the client.
Senior Member & Architect	Create all the required system design. Create High Level/ Low Level diagram.
Developers	Write codes based on the design provided by the senior member
Tester	Verify the codes written by developers is ready to be used in production

Table 60: Roles and Responsibilities in Waterfall Model (Shaikh, 2017)

7.6.4.3. Advantages of Waterfall

- The Waterfall model offers a clear and disciplined method to software development, providing a predictable output. (actiTIME, 2022)
- Comprehensive documentation is an essential part of the Waterfall model, enabling for future referencing and progress monitoring.
- The Waterfall model gives a great degree of control over the development process, allowing project managers to monitor progress, assure timely delivery, and remain below budget. (actiTIME, 2022)

7.7. Appendix I – Tools and Technology Used

[Return to top](#)

7.7.1. React JS

⇒ About React JS

ReactJS is a JavaScript library used to build powerful UI design. React JS is maintained by Meta. It is very powerful JS library with support of millions of developers. (Meta Platforms, 2022)

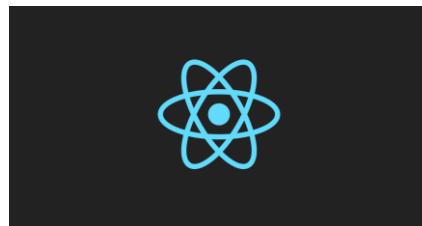


Figure 268:Logo of React JS

Source: (Meta Platforms, 2022)

⇒ React JS Use Case

React JS is used to create all the UI logic in the project. The library with help of other JS packages helps the project builds a powerful UI. The library with support of different hooks makes JavaScript easier.

⇒ Benefits of React JS

- Manage different change in UI without rendering.
- Achieve different JS logic easily using hooks.
- Use of JSX to build UI.

7.7.2. HTML / CSS (Tailwind CSS)

⇒ About HTML / CSS (Tailwind CSS)

HTML is a markup language used to define structure of the webpage. CSS is a stylesheet used to implement different design and style in the HTML structure. Likewise, Tailwind CSS is an open-source CSS framework allowing to style HTML using different class names.

⇒ HTML / CSS (Tailwind CSS) Use Case

All the above-mentioned topics are used to design and create structure of web application. Creating different forms adding design can be performed using above languages.

⇒ Benefits of using HTML / CSS (Tailwind CSS)

- Creating different structure of webpage easily
- Ability to apply style using only HTML file.
- No messy classes required.

7.7.3. Node JS

⇒ About Node JS

Node JS is an open-source JavaScript library that allows JS to load on the server side. With the development of Node JS JavaScript can be used as backend language as well. With NodeJS the processing can be perform asynchronous helping speed the performance. (w3schools, 2022)



Figure 269: Logo of NodeJS

Source: (NodeJS, 2022)

⇒ Node JS Use Case

In the project, the library is used as a backend language which helps to process different logic required. The library with help of express make the application building easier and faster.

⇒ Benefits of Node JS

- Huge packages available for different action
- Same language for both frontend and backend
- Better performance with asynchronous processing.

7.7.4. PostgreSQL

⇒ About PostgreSQL

With over 35 years of continuous development, the durable, open-source PostgreSQL object-relational database system has built a solid reputation for dependability, feature robustness, and performance. (PostgreSQL, 2022)



Figure 270: Logo of PostgreSQL (PostgreSQL, 2022)

Source: (PostgreSQL, 2022)

⇒ PostgreSQL Use Case

PostgreSQL is used database for the project. The database is used to store all the required data. All the action with the database is done directly by writing query in code.

⇒ Benefits of PostgreSQL

- Relational Database Benefits
- Advance Database functionality
- Easily integrate with macOS.

7.7.5. Elastic Search

⇒ About Elastic Search

Elastic search is an open-source search and analytics engine which allow to store, search, and analyze data stored in its database. It is a NoSQL database which stores data in format of JSON. It is famous for its fast ability to search through huge amount of data.

⇒ Elastic Search Use Case

Elastic Search is used as a search tool for the project. With its ability to search through data quickly required data are stored in elastic search cluster creating indexes. Different query and aggregation are then written to perform search.

⇒ Benefits of Elastic Search

- Faster search result
- Schema Free database
- Free to use.

7.8. SRS Document

7.8.1. Introduction

7.8.1.1. Purpose of Document

The SRS document represent all the requirement with a mutual understanding between both the client and the developers. The document helps to lay out all the requirement needed in a software and helps to define functionality of the feature. The document also includes all the required set of hardware and software required to complete the project ton its duration. The document is intended for both stakeholders and developers and must be approved by both parties.

7.8.1.2. Scope of the Document

The primary objective of the system is to build a web application, which has ability to move industry to digitalization phase, manage daily activities and help company enhanced their management system. The project is estimated to be completed within 120 business days. The product will also include a user guidebook to help new user understand the working of the system.

- Store data in digital format
- Manage different activities performed in the system,
- Allow user to view different report of sales and profit.
- Track and view balance sheet with its customers.
- Track and record attendance of employees.

7.8.1.3. Overview

The product being developed will be a web application containing required feature list to solve the problem listed by the client. The product will be developed with an aim to build a reliable, user-friendly, effective application for the client, which will not only solve the problem stated above but also help client be able to expand more path in business using the application. The product at the end will also help company view their daily activities by processing.

7.8.2. Overall Description

7.8.2.1. Project Perspective

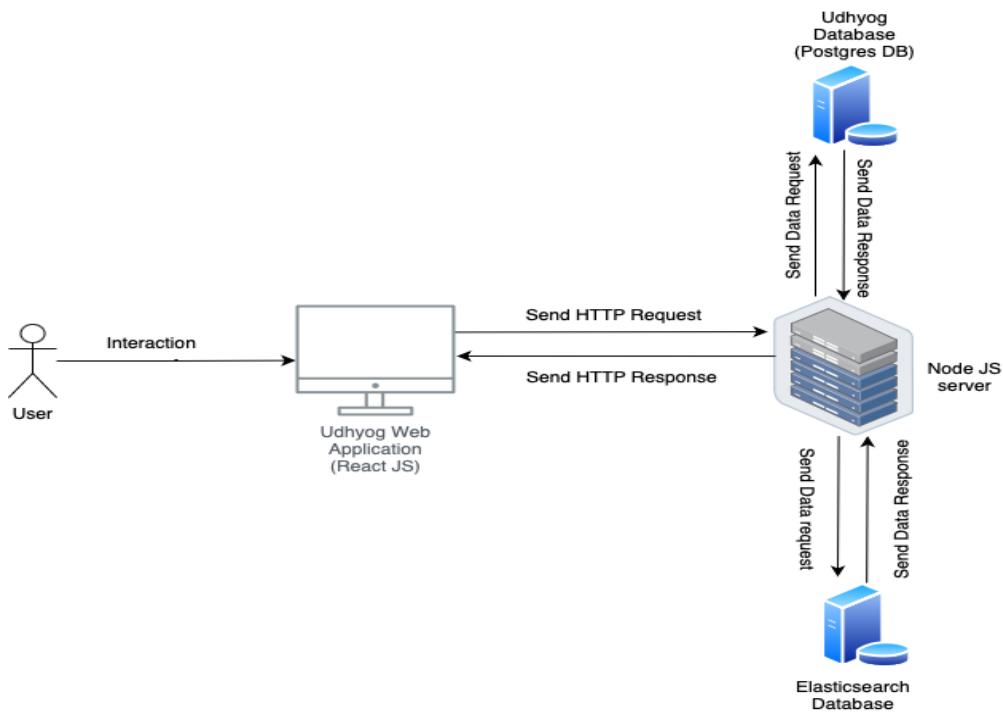


Figure 271: System Architecture of the application

The Udhyog application tends to replace the current manual process of workflow. With the use of application, it helps different manpower in factory to manage their daily task more efficiently.

7.8.2.2. User Classes and characteristics

Admin:

- Responsible for managing the user and assigning role to an individual.

Supervisor:

- Manage employees with the factory.
- Manage employee attendance.
- Bulk enters products and employees in the system.
- Export different reports

Staff:

- Create invoice using the application.
- Manage products in the application.
- Manage daily production information.

7.8.2.3. Operating environment

- The application shall operate on all modern browser (Chrome, Mozilla, Edge & Safari)
- The application must operate all task using GUI.
- The application must maintain secure accessibility.

7.8.2.4. Design and Implementation constraints

- Application will be developed using SCRUM methodology.
- Application shall operate on all modern browser (Chrome, Mozilla, Edge & Safari)
- The data will be stored in PostgreSQL.
- The application will be developed using React JS for frontend and Node JS for backend.

7.8.2.5. User Documentation

The user guide to application will be available within the application which will defiantly help new user understand working of the app.

7.8.2.6. System Features

7.8.2.6.1. Manage Staff and Supervisor

The web application allows admin to manage staff and supervisor in the system. The admin has ability to register a new staff and supervisor, deregister a staff and supervisor and change status of user to active and inactive state.

7.8.2.6.2. Manage Roles

The web application also allows admin to create new custom role which can be assigned to staff. The staff role can be customized with available role given to the staff. The role of supervisor and admin will be fixed.

7.8.2.6.3. Products in application

The application will also allow staff with role to add new products on the application. The staff are also responsible for editing the products and changing product status accordingly.

7.8.2.6.4. Employee in application

The supervisor of the system has ability to manage the employee of the factory. The supervisor can add new employee, edit employee information, change employee status, and add attendance of the employee.

7.8.2.6.5. Track Daily Product

With help of application user can easily insert daily production of each item. With this, application update the stock accordingly. The user can view and compare daily production using this data.

7.8.2.6.6. Manage Invoices

With help of application user can easily create invoice linking them to a customer. With this, application update the stock accordingly. The credit for the total will be added to customer ledger. The application also allows to edit the invoice after creating it, balance and inventory will be affected with the edit.

7.8.2.7. Functional Requirements

7.8.2.7.1. Register Staff / Supervisor

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-01	Admin can create a new user in the system.	
	SR-01	Admin request for registration form
	SR-02	Admin submit the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	The system checks email and username are unique or not
	SR-05	The system shows the message of whether user registration was successful or not

7.8.2.7.2. Login

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	User request for login form
	SR-02	Users submit the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	The system check whether submitted email/username and password matched with data in database.
	SR-05	The system throws error if login credentials were not correct
	SR-06	The redirect user to respective dashboard if user credentials were correct.

7.8.2.7.3. Add Role

Req.ID	Requirement Description	
FR-02	Sys.Req.ID	System Requirement
	SR-01	Admin request for role form
	SR-02	Users select required role and give a role name for the role
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	The system check whether the submitted role name is unique or not
	SR-05	The system throws error if role name already exists
	SR-06	The system sends success message if no error occurs

7.8.2.7.4. Edit Role

Req.ID	Requirement Description	
FR-02	Sys.Req.ID	System Requirement
	SR-01	Admin role form by sending id of role
	SR-02	System sends a form with data related to requested role id
	SR-03	Admin makes required changes and submit the form
	SR-04	The system check whether the submitted role name is unique or not
	SR-05	The system throws error if role name already exists
	SR-06	The system sends success message if no error occurs

7.8.2.7.5. Delete Role

Req.ID	Requirement Description	
FR-02	Admin has ability to delete the role assign to the staff.	
	Sys.Req.ID	System Requirement
	SR-01	Admin request to delete role from the system
	SR-02	System request for confirmation of the requested action
	SR-03	Admin confirms the request from confirmation box
	SR-04	If admin confirms the request system delete the role and checks if role is current assigned to user.
SR-05	If role is assigned to user, system set user's role to null	

7.8.2.7.6. Add Product

Req.ID	Requirement Description	
FR-02	The user with role of staff can add a new product in the system.	
	Sys.Req.ID	System Requirement
	SR-01	Staff request a form to add new product
	SR-02	Staff send the form with all the required details in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	The system check whether the submitted product ID is unique or not
	SR-05	If system validation succeeds, system add the product to system and notify user
SR-06	If system validation fails, system notify user about the error	

7.8.2.7.7. Edit Product

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Staff request a form to edit product by send id
	SR-02	System sends the form with sorted data
	SR-02	Staff send the form with all the required edits in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	The system check whether the submitted product ID is unique or not
	SR-05	If system validation succeeds, system edit the product to system and notify user
	SR-06	If system validation fails, system notify user about the error

7.8.2.7.8. Create Invoice

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Staff request a form to add new invoice
	SR-02	Staff send the form with all the required details in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	If system validation succeeds, system add the product to system and notify user
	SR-05	The system adds the grand total of the invoice to the customer ledger and reduce the product quantity respective to number of products added in the invoice

7.8.2.7.9. Edit Invoice

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Staff request a form to edit invoice with invoice id
	SR-02	System sends the form with sorted data
	SR-02	Staff send the form with all the required edits in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	If system validation succeeds, system add the product to system and notify user
	SR-05	The system adjusts the grand total of the invoice to the customer ledger and adjust the product quantity respective to number of products added in the invoice

7.8.2.7.10. Add Customers

Req.ID	Requirement Description	
FR-02	The user with role of staff can add a customer in the system.	
	Sys.Req.ID	System Requirement
	SR-01	Staff request a form to add new customer
	SR-02	Staff send the form with all the required details in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	If system validation succeeds, system add the product to system and notify user

7.8.2.7.11. Edit Customer

Req.ID	Requirement Description	
FR-02	The user with role of staff can edit customer in the system.	
	Sys.Req.ID	System Requirement
	SR-01	Staff request a form to edit customer by send id
	SR-02	System sends the form with sorted data
	SR-02	Staff send the form with all the required edits in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-05	If system validation succeeds, system edit the product to system and notify user

7.8.2.7.12. Add Money Transaction

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Staff request a form to add new money transaction
	SR-02	Staff send the form with all the required details in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	If system validation succeeds, system add the product to system and notify user
	SR-05	The system adds the transaction to customer ledger

7.8.2.7.13. Add Employee

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Supervisors request a form to add new employee
	SR-02	Supervisors send the form with all the required details in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-04	If system validation succeeds, system add the employee to system and notify supervisor

7.8.2.7.14. Edit Employee

Req.ID	Requirement Description	
	Sys.Req.ID	System Requirement
FR-02	SR-01	Supervisors request a form to edit employee by send id
	SR-02	System sends the form with sorted data
	SR-02	Staff send the form with all the required edits in the form
	SR-03	The system verifies all the fields and throw error if data cannot be validated
	SR-05	If system validation succeeds, system edit the product to system and notify user

7.8.2.8. Other Non-Functional Requirements

7.8.2.8.1. Quality Requirements

Req-ID	Requirement Description	Priority
QR-01	The application should run smoothly in all modern browser	Should
QR-02	The application must be hosted in high-performance server	Should
QR-03	The application could have a pleasing UI design	Could
QR-04	The user could use high-speed internet	Could

7.8.2.8.2. Security Requirements

Req-ID	Requirement Description	Priority
SR-01	The security of server should be managed by the client	Should
SR-02	The application should not contain any third-party tracking to effect user experience	Should
SR-03	The application should use secure way to save user password in database	Should
SR-04	Any secured content could be communicated in encrypted format	Could

7.8.2.8.3. Maintainability Requirements

Req-ID	Requirement Description	Priority
MR-01	The application should continue update on application in future for bugs fix	Should
MR-02	The application could add new features as per requirements	Could

7.9. Important Coding Logic

7.9.1. Cron Jobs In Application



```
1 const eCron = nodeCron.schedule('0 9 * * 6', cron.addEmployeeAttendance)
2 const overdueCron = nodeCron.schedule('0 9 * * 6', cron.getInvoicesWithAmounts)
3 const lowStockCron = nodeCron.schedule('0 9 * * *', cron.lowStockProduct)
4 eCron.start()
5 overdueCron.start()
6 lowStockCron.start()
```

Figure 272: Screenshot of calling cron job.



```
1 module.exports.addEmployeeAttendance = async () => {
2     console.log("Runned")
3     const getActiveEmployee = await custom.customPromise(`SELECT id FROM employees
4 WHERE status='active'`)
5     console.log(getActiveEmployee)
6     if (getActiveEmployee.length == 0) {
7         insertLog("Employee Attendance", true, '')
8     } else {
9         const activeEmployee = []
10        getActiveEmployee?.forEach(x => {
11            activeEmployee.push([
12                x.id,
13                adToBs(dateFormat(new Date(), 'yyyy-MM-dd')),
14                "present",
15            ])
16        })
17        const query = `INSERT INTO employee_attendance(employee,date,attendance) VALUES %L`
18        db.query(format(query, activeEmployee), [], (err, result) => {
19            if (err) {
20                console.log(err)
21                insertLog('Employee Attendance', false, err)
22            } else {
23                insertLog('Employee Attendance', true, '')
24            }
25        })
26    }
27    return false
28 }
```

Figure 273: Screenshot of adding employee attendance.



```
1 module.exports.lowStockProduct = () => {
2     try {
3         notification.getNotification()
4         insertLog('Low Stock Alert', true, '')
5     } catch (err) {
6         insertLog('Low Stock Alert', false, 'Unknown error occurred');
7     }
8 }
9
10 }
```

Figure 274: Screenshot of getting low stock notification.



```
1 const insertLog = (logTitle, status, description) => {
2     return new Promise((resolve) => {
3         const query = `INSERT INTO cron_log(date,log,status,description) VALUES ($1,$2,$3,$4)`
4         db.query(query, [adToBs(dateFormat(new Date(), 'yyyy-MM-dd')), logTitle, status, description], (err, result) => {
5             if (err) {
6                 console.log(err)
7                 resolve({ success: false })
8             } else {
9                 resolve({ success: true })
10            }
11        })
12    })
13 }
```

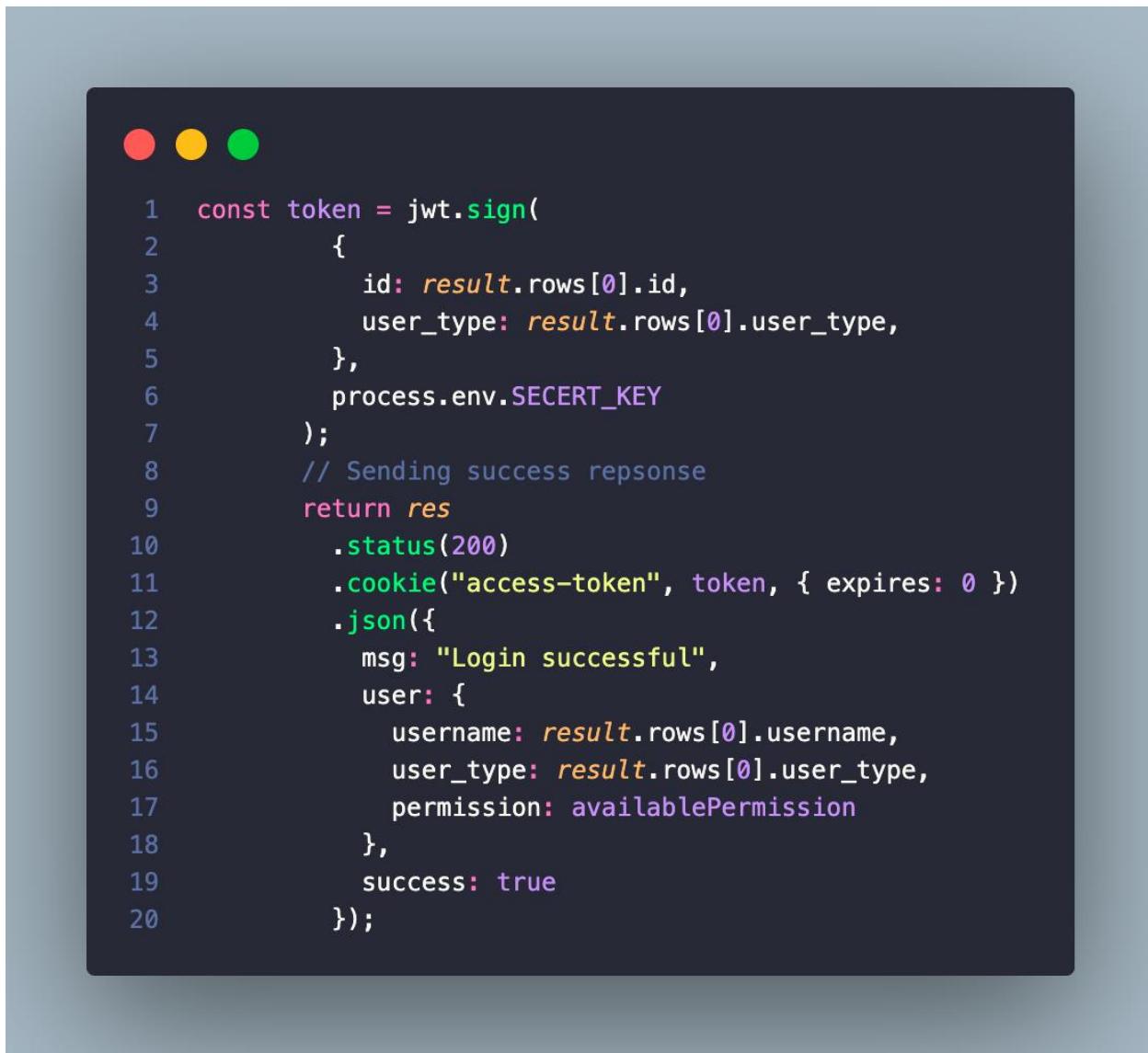
Figure 275: Screenshot of maintaining cron log.

7.9.2. Global Mail Function



```
1  async function sendMail(to, subject, data, file, attachments) {
2    // create reusable transporter object using the default SMTP transport
3    try {
4      const transporter = nodemailer.createTransport({
5        host: "smtp.ethereal.email",
6        port: 587,
7        auth: {
8          user: "antonia41@ethereal.email",
9          pass: "q8qFXHTaNGSuzQwAhv",
10        },
11      });
12      // use a template file with nodemailer
13      transporter.use(
14        "compile",
15        hbs({
16          viewEngine: {
17            extname: ".handlebars",
18            layoutsDir: "./mail/",
19            defaultLayout: false,
20            partialsDir: "./mail/",
21          },
22          viewPath: "./mail/",
23          extName: ".handlebars",
24        })
25      );
26      transporter.sendMail(
27        {
28          from: "bert.bayer74@ethereal.email",
29          to: to,
30          subject: subject,
31          template: file,
32          context: data,
33          attachments: attachments,
34        },
35        (error, info) => {
36          if (error) {
37            console.log(error);
38          } else {
39            console.log("Preview URL: " + nodemailer.getTestMessageUrl(info));
40            console.log("Mail send");
41          }
42        }
43      );
44    } catch (err) {
45      console.log(err);
46    }
47  }
48
49 module.exports = { sendMail };
```

Figure 276: Screenshot of global mail function

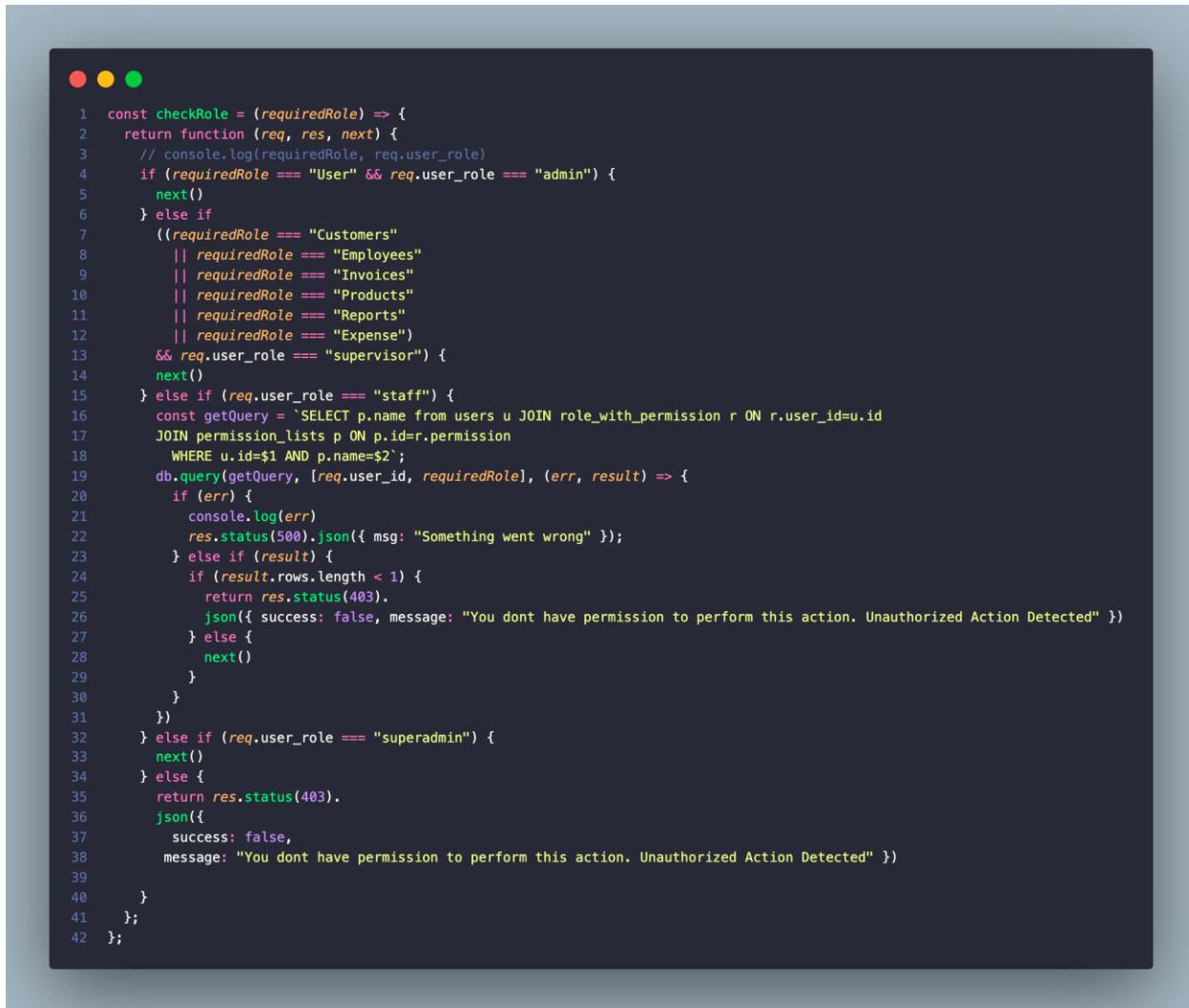


A screenshot of a terminal window with three colored window control buttons (red, yellow, green) at the top. The terminal displays a block of Node.js code. The code uses the `jwt` library to sign a token with user information, then sends a success response with a cookie and JSON data.

```
1 const token = jwt.sign(
2   {
3     id: result.rows[0].id,
4     user_type: result.rows[0].user_type,
5   },
6   process.env.SECERT_KEY
7 );
8 // Sending success repsonse
9 return res
10   .status(200)
11   .cookie("access-token", token, { expires: 0 })
12   .json({
13     msg: "Login successful",
14     user: {
15       username: result.rows[0].username,
16       user_type: result.rows[0].user_type,
17       permission: availablePermission
18     },
19     success: true
20   });

```

Figure 277: Cookie session setup



```
1 const checkRole = (requiredRole) => {
2   return function (req, res, next) {
3     // console.log(requiredRole, req.user_role)
4     if (requiredRole === "User" && req.user_role === "admin") {
5       next()
6     } else if (
7       ((requiredRole === "Customers"
8         || requiredRole === "Employees"
9         || requiredRole === "Invoices"
10        || requiredRole === "Products"
11        || requiredRole === "Reports"
12        || requiredRole === "Expense")
13       && req.user_role === "supervisor") {
14       next()
15     } else if (req.user_role === "staff") {
16       const getQuery = `SELECT p.name from users u JOIN role_with_permission r ON r.user_id=u.id
17       JOIN permission_lists p ON p.id=r.permission
18       WHERE u.id=$1 AND p.name=$2`;
19       db.query(getQuery, [req.user_id, requiredRole], (err, result) => {
20         if (err) {
21           console.log(err)
22           res.status(500).json({ msg: "Something went wrong" });
23         } else if (result) {
24           if (result.rows.length < 1) {
25             return res.status(403).
26             json({ success: false, message: "You dont have permission to perform this action. Unauthorized Action Detected" })
27           } else {
28             next()
29           }
30         }
31       })
32     } else if (req.user_role === "superadmin") {
33       next()
34     } else {
35       return res.status(403).
36       json({
37         success: false,
38         message: "You dont have permission to perform this action. Unauthorized Action Detected"
39       })
40     }
41   };
42 };
```

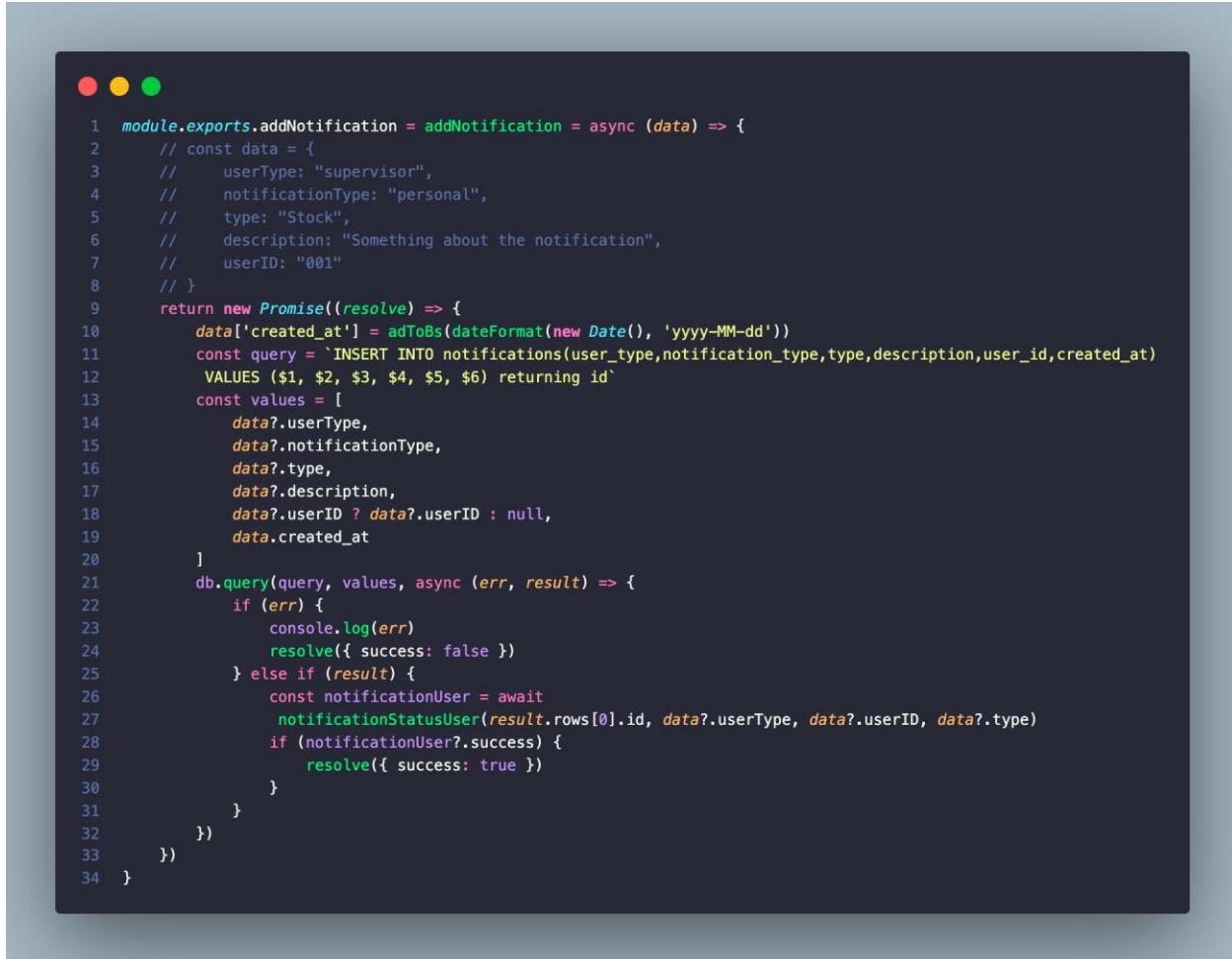
Figure 278: Screenshot of check role middleware



A screenshot of a code editor window titled "File" with three tabs at the top. The main area contains the following code:

```
1 const storage = multer.diskStorage({
2   destination: (req, file, callback) => {
3     // save file location
4     callback(null, "./public/images");
5   },
6   // adding file name
7   filename: (req, file, callback) => {
8     callback(null, Date.now() + path.extname(file.originalname));
9   },
10 });
11 // upload file
12 const upload = multer({ storage: storage });
13
14 module.exports = { upload };
15
```

Figure 279: Screenshot of file upload middleware



```
1 module.exports.addNotification = addNotification = async (data) => {
2     // const data = {
3     //     userType: "supervisor",
4     //     notificationType: "personal",
5     //     type: "Stock",
6     //     description: "Something about the notification",
7     //     userID: "001"
8     // }
9     return new Promise((resolve) => {
10         data['created_at'] = adToBs(dateFormat(new Date(), 'yyyy-MM-dd'))
11         const query = `INSERT INTO notifications(user_type,notification_type,type,description,user_id,created_at)
12             VALUES ($1, $2, $3, $4, $5, $6) returning id`
13         const values = [
14             data?.userType,
15             data?.notificationType,
16             data?.type,
17             data?.description,
18             data?.userID ? data?.userID : null,
19             data.created_at
20         ]
21         db.query(query, values, async (err, result) => {
22             if (err) {
23                 console.log(err)
24                 resolve({ success: false })
25             } else if (result) {
26                 const notificationUser = await
27                     notificationStatusUser(result.rows[0].id, data?.userType, data?.userID, data?.type)
28                 if (notificationUser?.success) {
29                     resolve({ success: true })
30                 }
31             }
32         })
33     })
34 }
```

Figure 280: Screenshot of global user notification



```

1  await client.index(
2      {
3          index: "employees",
4          id: `${result.rows[0].id}`,
5          body: {
6              name: `${req.body.name}`,
7              phone: `${req.body.phone}`,
8              location: `${req.body.location}`,
9              pan: `${req.body.pan}`,
10             joined_date: `${req.body.joined_date}`,
11             is_active: `${req.body.status ? req.body.status : 'active'}`,
12         },
13     },
14     (err) => {
15         {
16             err && console.log(err);
17         }
18     }
19 );

```

Figure 281: Screenshot of adding data to elastic search.



```

1 const createInvoicePDF = (invoiceDetails, path) => {
2     try {
3         let doc = new PDFDocument({ margin: 50 });
4         generateHeader(doc);
5         generatePDFHeading(doc, "Invoice");
6         generatePDFDate(doc, invoiceDetails?.date);
7         generateCustomer(doc, invoiceDetails?.customer_name, invoiceDetails.customer_location);
8         generateInvoiceFullTable(doc, invoiceDetails?.items, invoiceDetails?.subTotal, invoiceDetails?.discount, invoiceDetails?.vat, invoiceDetails?.grandTotal)
9         // generateOverdueTable(doc, overdue);
10        doc.pipe(fs.createWriteStream(path));
11        doc.flushPages();
12        doc.end();
13        console.log(`PDF invoice saved to ${path}`);
14    } catch (err) {
15        console.log(err);
16    }
17 };

```

Figure 282: Screenshot of function creating invoice PDF

7.10. Git according to Sprint

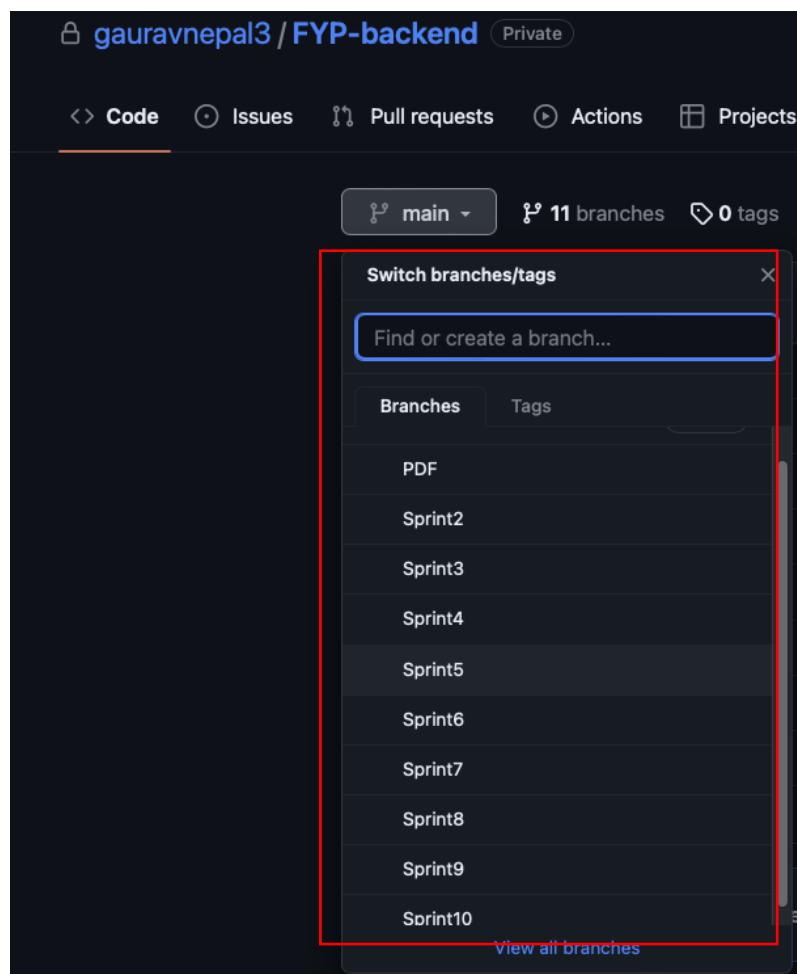


Figure 283: Screenshot of branch maintained in Git.

7.11. Originality Report

4/8/23, 9:29 PM

Originality report

COURSE NAME
CS6P05 - FYP - Prithivi Maharjan and Ankit Chhetri

STUDENT NAME
GAURAV KUMAR NEPAL

FILE NAME
GAURAV KUMAR NEPAL - Final Year Final Report Draft (3/3)

REPORT CREATED
8 Apr 2023

Summary

Flagged passages	3	0.5%
Cited/quoted passages	0	0%

Web matches

vyaparapp.in	1	0.3%
axureboutique.com	1	0.1%
minitab.com	1	0.1%

1 of 3 passages
Student passage FLAGGED

Vyapar is a FREE Business Accounting Software designed for Indian small businesses to handle invoicing, inventory, accounting, and other tasks. The idea is to make a businessman's daily routine less...

Top web match

Vyapar is a FREE Business Accounting Software built for Indian small businesses to deal with invoicing, inventory, accounting needs, and much more! The goal is to make a businessman's daily routine...

About Vyapar App a Billing Software <https://vyaparapp.in/gst-billing-app>

about:blank

Page 1 of 2

Figure 284: Originality report