

Application Development

Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS			
Programme title	BITEC Higher National Diploma in Computing		
Assessor		Internal Verifier	
Unit(s)	Unit 30 – Application Development		
Assignment title	Inventory Control Application for BAUHINIA		
Student's name	Ryan Wickramaratne (COL 00081762)		
List which assessment criteria the Assessor has awarded.	Pass	Merit	Distinction
INTERNAL VERIFIER CHECKLIST			
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N		
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N		
Has the work been assessed accurately?	Y/N		
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	Y/N Y/N Y/N Y/N		
Does the assessment decision need amending?	Y/N		
Assessor signature			Date
Internal Verifier signature			Date
Programme Leader signature (if required)			Date

Confirm action completed			
Remedial action taken Give details:			
Assessor signature			Date
Internal Verifier signature			Date



Programme Leader signature (if required)		Date	
---	--	-------------	--



Higher Nationals - Summative Assignment Feedback Form

Student Name/ID	Ryan Wickramaratne (COL 00081762)		
Unit Title	Unit 30 – Application Development		
Assignment Number	1	Assessor	
Submission Date	20/07/2023	Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	

Assessor Feedback:

LO1 Produce a Software Design Document by analysing a business-related problem and deduce an appropriate solution including a set of initial requirements

Pass, Merit & Distinction P1 P2 M1
Descripts

LO2 Use design and development methodologies with tools and techniques associated with the creation of a business application

Pass, Merit & Distinction P3 M2 D1
Descripts

LO3 Work individually and as part of a team to plan and produce a functional business application with support documentation

Pass, Merit & Distinction P4 P5 M3 M4 D2
Descripts

LO4 Evaluate the performance of a business application against its Software Design Document and initial requirements

Pass, Merit & Distinction P6 M5 D3
Descripts

Grade:	Assessor Signature:	Date:
--------	---------------------	-------

Resubmission Feedback:

Grade:	Assessor Signature:	Date:
--------	---------------------	-------

Internal Verifier's Comments:

Signature & Date:

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.



Assignment Feedback

Formative Feedback: Assessor to Student

Action Plan

Summative feedback

Feedback: Student to Assessor



Assessor signature		Date	
Student signature		Date	



Pearson Higher Nationals in Computing

Unit 30 – Application Development

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing.** Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each page.** This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

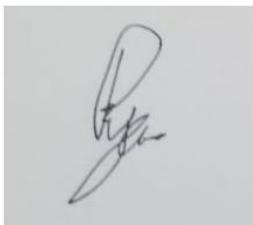
Important Points:

1. **It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.**
2. Avoid using page borders in your assignment body.
3. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
4. Ensure that you give yourself enough time to complete the assignment by the due date.
5. Excuses of any nature will not be accepted for failure to hand in the work on time.
6. You must take responsibility for managing your own time effectively.
7. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
8. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
9. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
10. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
11. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct form. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of Edexcel UK.
3. I know what the consequences will be if I plagiarise or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspect of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Edexcel UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the assignment.



20/07/2023

ryandilthusha@gmail.com

Student's Signature:
(Provide E-mail ID)

Date:
(Provide Submission Date)



Higher National Diploma in Business



Assignment Brief

Student Name /ID Number	Ryan Wickramaratne (COL 00081762)
Unit Number and Title	Unit 30: Application Development
Academic Year	2021/22
Unit Tutor	Mr. Dileepa
Assignment Title	Application Development
Issue Date	
Submission Date	20/07/2023
IV Name & Date	

Submission format

The submission should be in the form of an individual written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research. You must provide in-text citations and the reference list using Harvard referencing system.

The recommended word count is 4,000–4,500 words excluding annexures..

Minimum word count – 4,000

Maximum word count – 4,500

Unit Learning Outcomes:

LO1 Produce a Software Design Document by analysing a business-related problem and deduce an appropriate solution including a set of initial requirements.

LO2 Use design and development methodologies with tools and techniques associated with the creation of a business application.

LO3 Work individually and as part of a team to plan and produce a functional business application with support documentation.

LO4 Evaluate the performance of a business application against its Software Design Document and initial requirements

Assignment Brief and Guidance:

BAUHINIA is a clothing brand in Sri Lanka, founded in 2018, which has come a long way, offering Sri Lankans with great designs of a variety of clothing. Currently, BAUHINIA is handling orders through social media networks such as Facebook and Instagram. Customers can message BAUHINIA requesting an item/s by sending the item code, size and required quantity. If the item is available, the customer is required to send the delivery address, contact number to confirm the order. The package with the required item/s will be delivered to the customer's doorstep within 3 to 5 working days, after which he/she must pay cash on delivery.

Over the years, BAUHINIA has grown steadily mainly due to its popularity over social media. However, they are finding it increasingly difficult to cope up with paperwork associated with inventory management due to the increased of number of orders through message requests. The Managing Director is frustrated by the problems associated with inventory management and has decided that BAUHINIA will consult a Software Development Company to automate the workflow of BAUHINIA Clothing.

AKL Software (AKL) is a software development consultant. AKL has purpose-built rooms for Facilitated Workshops and Agile software development projects. The Managing Director of BAUHINIA has decided to contract AKL for the development of the new order tracking system using an Agile development approach.

The new online solution will replace the old approach and is likely to include some of the following functionality:

- Customer Registration and sign-in : Allow customers to register free. At the time of registration, customers need to provide name, email address, delivery address, password and two working telephone numbers. Registered customers can sign-in using email address and password.
- Browse for products: through product catalog, check availability and add products to cart.
- Checkout products: Total amount to be paid will be shown. Customer will be redirected to confirm billing details: Name, delivery address, email address, two contact numbers. Payment method will be cash on delivery.
- Staff registration and sign-in.
- Create a daily report of orders that have been requested – carried out by the Production Manager.
- Create a daily report of product availability- carried out by Production Manager.

- Add new items to inventory, update existing item details – carried out by Inventory handling Clerk.
- Create a monthly Income report- carried out of chief Accountant.

The new online solution should have the following levels of access:

- Report only
- Update only
- Update and delete
- Complete system access

Activity 1

1.1 Produce a well-defined Problem definition statement supported by a set of user and System requirements for the above scenario. Identify areas (if any) of risk that might affect the successful completion of the application.

1.2 Produce a well-structured Software Design Document that defines a proposed solution for BAUHINIA by exploring and analyzing their business problem. Include relevant details on requirements, system analysis, system design. (propose a suitable language)

Activity 2

Investigate the use of software development tools and techniques for the chosen software solution. Compare the investigated tools and techniques and justify the chosen tools, technique and methodology that you may use for the development of an Inventory Control Application for BAUHINIA .

Activity 3

3.1 Create a presentation to review followings;

- Business application
- Problem definition statement
- Proposed solution
- Development strategy

Conduct a peer-review and identify opportunities that were not previously considered by interpreting the recorded feedback.

3.2 Develop a business application with support documentation, based on the Software Design Document produced in activity 1, along with supportive evidence for using the preferred tools, techniques and methodologies investigated in activity 2. Assess new ideas or possible improvements to the system developed while justifying the reasons for including/not including them in the application developed.

Activity 4

Conduct a critical review for the design, development, and testing stages of the Inventory Control application by analyzing the factors that influence its performance against the problem definition statement and initial requirements. Conclude the review by reflectively discussing the risks identified at the beginning and critically evaluating the strengths and weaknesses of the application developed. Identify and justify the opportunities for improvement and further development of the application you developed.

Grading Criteria	Achieved	
LO1 Produce a Software Design Document by analysing a business-related problem and deduce an appropriate solution including a set of initial requirements.		
P1 Explore a business related problem and produce a well-defined Problem Definition Statement supported by a set of user and system requirements.		
P2 Determine any areas of risk related to the successful completion of your application.		
M1 Analyse a business related problem using appropriate methods and produce a well-structured Software Design Document that defines a proposed solution and includes relevant details on requirements, system analysis, system design, coding, testing and implementation.		
LO2 Use design and development methodologies with tools and techniques associated with the creation of a business application.		
P3 Research the use of software development tools and techniques and identify any that have been selected for the development of this application.		
M2 Compare the differences between the various software development tools and techniques researched and justify your preferred selection as well as your preferred software development methodology.		
LO1 & LO2 Justify your solution to a business-related problem and your preferred software development methodology, by comparing between the various software development tools and techniques researched.		
LO3 Work individually and as part of a team to plan and produce a functional business application with support documentation.		

P4 Create a formal presentation that effectively reviews your business application, problem definition statement, proposed solution and development strategy. Use this presentation as part of a peer-review and document any feedback given.		
P5 Develop a functional business application with support documentation based on a specified business problem.		
M3 Interpret your peer review feedback and identify opportunities not previously considered.		
M4 Develop a functional business application based on a specific Software Design Document with supportive evidence of using the preferred tools, techniques and methodologies.		
D2 Evaluate any new insights, ideas or potential improvements to your system and justify the reasons why you have chosen to include (or not to include) them as part of this business application.		
LO4 Evaluate the performance of a business application against its Software Design Document and initial requirements.		
P6 Review the performance of your business application against the Problem Definition Statement and initial requirements.		
M5 Analyse the factors that influence the performance of a business application and use them to undertake a critical review of the design, development, and testing stages of your application. Conclude your review by reflectively discussing your previously identified risks.		
D3 Critically evaluate the strengths and weaknesses of your business application and fully justify opportunities for improvement and further development.		

Acknowledgement

I would like to express my special thanks of gratitude to my AD lecturer Mr. Dileepa for providing invaluable guidance and giving immense amount of knowledge to work on this assignment perfectly. I specially thanks him because he helped us in doing a lot of research and I came to know about so many new things about the application developing techniques.

Secondly, I would like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Executive Summary

In Activity 1, I examined the problems of the current inventory system, defined user and system requirements, and outlined potential risks. I then created a detailed Software Design Document (SDD) that included comprehensive system design, coding standards, and risk mitigation strategies.

Activity 2 involved the selection of software development tools and techniques, with a focus on Agile and Scrum methodologies.

In Activity 3, I developed and reviewed the BAUHINIA project. Peer reviews, system requirements alignment, and comprehensive documentation were key components of this activity. I also proposed future improvements, including advanced analytics, AI integration, and user customization.

Finally, Activity 4 provided a critical review of the project, assessing the design, development, testing stages, and overall performance of the business application. This stage also revisited the risk management process and suggested areas for future enhancements. The project offered valuable learnings and established a solid groundwork for the further development of the BAUHINIA Inventory Control Application.

List of figures

FIGURE 1. 1 ERD DIAGRAM	82
FIGURE 1. 2 CLASS DIAGRAM	84
FIGURE 1. 3 USE CASE DIAGRAM	87
FIGURE 1. 4 SCENARIO 1: CUSTOMER PLACES AN ORDER	90
FIGURE 1. 5 SCENARIO 2: STAFF UPDATES INVENTORY	92
FIGURE 1. 6 SCENARIO 3: CUSTOMER TRACKS AN ORDER	94
FIGURE 1. 7 SCENARIO 4: STAFF GENERATES REPORT.....	96
FIGURE 1. 8 CUSTOMER - LOGIN PAGE	101
FIGURE 1. 9 CUSTOMER - SIGN-UP PAGE	101
FIGURE 1. 10 CUSTOMER - HOME PAGE.....	103
FIGURE 1. 11 CUSTOMER – PRODUCTS BROWSE PAGE.....	104
FIGURE 1. 12 CUSTOMER – PRODUCTS DETAILS PAGE.....	105
FIGURE 1. 13 CUSTOMER – CHECKOUT PAGE	106
FIGURE 1. 14 CUSTOMER – SHOPPING CART PAGE.....	107
FIGURE 1. 15 CUSTOMER – ORDER TRACKING PAGE	108
FIGURE 1. 16 STAFF – LOGIN PAGE	109
FIGURE 1. 17 STAFF – SIGN-UP PAGE	109
FIGURE 1. 18 STAFF – REPORTS PAGE.....	110
FIGURE 1. 19 STAFF – INVENTORY UPDATE PAGE.....	111
FIGURE 1. 20 STAFF – DASHBOARD PAGE	112
FIGURE 1. 21 LOGICAL DATABASE DIAGRAM DESIGN	113
FIGURE 3. 1 SLIDE 1.....	284
FIGURE 3. 2 SLIDE 2.....	284
FIGURE 3. 3 SLIDE 3.....	285
FIGURE 3. 4 SLIDE 4.....	285
FIGURE 3. 5 SLIDE 5.....	286
FIGURE 3. 6 SLIDE 6.....	286
FIGURE 3. 7 SLIDE 7.....	287
FIGURE 3. 8 SLIDE 8.....	287
FIGURE 3. 9 SLIDE 9.....	288
FIGURE 3. 10 SLIDE 10.....	288
FIGURE 3. 11 SLIDE 11	289
FIGURE 3. 12 SLIDE 12	289
FIGURE 3. 13 SLIDE 13.....	290
FIGURE 3. 14 SLIDE 14.....	290
FIGURE 3. 15 SLIDE 15.....	291
FIGURE 3. 16 SLIDE 16.....	291
FIGURE 3. 17 SLIDE 17.....	292
FIGURE 3. 18 SLIDE 18.....	292
FIGURE 3. 19 SLIDE 19.....	293
FIGURE 3. 20 SLIDE 20.....	293
FIGURE 3. 21 SLIDE 21.....	294
FIGURE 3. 22 SLIDE 22.....	294
FIGURE 3. 23 SLIDE 23.....	295

FIGURE 3. 24 SLIDE 24	295
FIGURE 3. 25 SLIDE 25	296
FIGURE 3. 26 SLIDE 26	296
FIGURE 3. 27 SLIDE 27	297
FIGURE 3. 28 SLIDE 28	297
FIGURE 3. 29 SLIDE 29	298
FIGURE 3. 30 SLIDE 30	298
FIGURE 3. 31 SLIDE 31	299
FIGURE 3. 32 SLIDE 32	299
FIGURE 3. 33 SLIDE 33	300
FIGURE 3. 34 SLIDE 34	300
FIGURE 3. 35 SLIDE 35	301
FIGURE 3. 36 SLIDE 36	301
FIGURE 3. 37 SLIDE 37	302
FIGURE 3. 38 SLIDE 38	302
FIGURE 3. 39 SLIDE 39	303
FIGURE 3. 40 SLIDE 40	303
FIGURE 3. 41 SLIDE 41	304
FIGURE 3. 42 SLIDE 42	304
FIGURE 3. 43 SLIDE 43	305
FIGURE 3. 44 SLIDE 44	305
FIGURE 3. 45 SLIDE 45	306
FIGURE 3. 46 SLIDE 46	306
FIGURE 3. 47 SLIDE 47	307
FIGURE 3. 48 SLIDE 48	307
FIGURE 3. 49 SLIDE 49	308
FIGURE 3. 50 ENTITY RELATIONSHIP DIAGRAM (ERD):	332
FIGURE 3. 51 CLASS DIAGRAM:	333
FIGURE 3. 52 USE CASE DIAGRAM:	334
FIGURE 3. 53 LOGICAL DATABASE DIAGRAM:	335
FIGURE 3. 54 SCENARIO 1: CUSTOMER PLACES AN ORDER	336
FIGURE 3. 55 SCENARIO 2: STAFF UPDATES INVENTORY	336
FIGURE 3. 56 SCENARIO 3: CUSTOMER TRACKS AN ORDER	337
FIGURE 3. 57 SCENARIO 4: STAFF GENERATES REPORT	338
FIGURE 3. 58 CUSTOMER – LOGIN PAGE	340
FIGURE 3. 59 CUSTOMER – SIGN-UP PAGE	340
FIGURE 3. 60 CUSTOMER – HOME PAGE	341
FIGURE 3. 61 CUSTOMER – PRODUCTS BROWSE PAGE	342
FIGURE 3. 62 CUSTOMER – PRODUCTS DETAILS PAGE	343
FIGURE 3. 63 CUSTOMER – CHECKOUT PAGE	344
FIGURE 3. 64 CUSTOMER – SHOPPING CART PAGE	345
FIGURE 3. 65 CUSTOMER – ORDER TRACKING PAGE	345
FIGURE 3. 66 STAFF – LOGIN PAGE	346
FIGURE 3. 67 STAFF – SIGN-UP PAGE	346
FIGURE 3. 68 STAFF – REPORTS PAGE	347
FIGURE 3. 69 STAFF – INVENTORY UPDATE PAGE	347
FIGURE 3. 70 STAFF – DASHBOARD PAGE	348
FIGURE 3. 71 CODE EXAMPLE	359

List of Tables

TABLE 1. 1 RISK REPORT FORM – FOR TECHNICAL RISKS	43
TABLE 1. 2 RISK REPORT FORM – FOR PROJECT MANAGEMENT RISKS.....	45
TABLE 1. 3 RISK REPORT FORM – FOR FINANCIAL RISKS.....	46
TABLE 1. 4 RISK REPORT FORM – FOR OPERATIONAL RISKS.....	47
TABLE 1. 5 RISK REPORT FORM – FOR LEGAL AND COMPLIANCE RISKS	48
TABLE 1. 6 RISK REPORT FORM – FOR SECURITY RISKS	49
TABLE 1. 7 RISK REPORT FORM – FOR USABILITY RISKS	50
TABLE 1. 8 RISK REPORT FORM – FOR VENDOR AND THIRD-PARTY RISKS	51
TABLE 1. 9 RISK REPORT FORM – FOR ORGANIZATIONAL CHANGE RISKS.....	52
TABLE 1. 10 RISK REPORT FORM – FOR SCALABILITY RISKS.....	53
TABLE 1. 11 FUNCTIONAL REQUIREMENTS	66
TABLE 1. 12 NON-FUNCTIONAL REQUIREMENT.....	67
TABLE 1. 13 TEST CASE 1.1: CUSTOMER SIGN-UP	137
TABLE 1. 14 TEST CASE 1.2: STAFF SIGN-UP	138
TABLE 1. 15 TEST CASE 2.1: CUSTOMER LOGIN	139
TABLE 1. 16 TEST CASE 2.2: STAFF LOGIN	140
TABLE 1. 17 TEST CASE 3: PRODUCT BROWSING FUNCTIONALITY.....	141
TABLE 1. 18 TEST CASE 4: PRODUCT DETAILS PAGE FUNCTIONALITY.....	142
TABLE 1. 19 TEST CASE 5: ADD TO CART AND MODIFY CART FUNCTIONALITY.....	143
TABLE 1. 20 TEST CASE 6: CHECKOUT PROCESS	144
TABLE 1. 21 TEST CASE 7: ORDER TRACKING FUNCTIONALITY FOR CUSTOMERS	146
TABLE 1. 22 TEST CASE 8: REPORT GENERATION FUNCTIONALITY FOR STAFF.....	147
TABLE 1. 23 TEST CASE 9: INVENTORY UPDATE FUNCTIONALITY FOR STAFF.....	148
TABLE 1. 24 TEST CASE 10: DASHBOARD FUNCTIONALITY FOR BOTH CUSTOMERS AND STAFF.....	149
TABLE 1. 25 RISK REPORT FORM – FOR TECHNICAL RISKS	162
TABLE 1. 26 RISK REPORT FORM – FOR PROJECT MANAGEMENT RISKS	164
TABLE 1. 27 RISK REPORT FORM – FOR FINANCIAL RISKS.....	165
TABLE 1. 28 RISK REPORT FORM – FOR OPERATIONAL RISKS.....	166
TABLE 1. 29 RISK REPORT FORM – FOR LEGAL AND COMPLIANCE RISKS	167
TABLE 1. 30 RISK REPORT FORM – FOR SECURITY RISKS	168
TABLE 1. 31 RISK REPORT FORM – FOR USABILITY RISKS	169
TABLE 1. 32 RISK REPORT FORM – FOR VENDOR AND THIRD-PARTY RISKS	170
TABLE 1. 33 RISK REPORT FORM – FOR ORGANIZATIONAL CHANGE RISKS.....	171
TABLE 1. 34 RISK REPORT FORM – FOR SCALABILITY RISKS.....	172
 TABLE 2. 1 COMPARISON DIFFERENCES BETWEEN EACH SOFTWARE DEVELOPMENT TOOLS	244
TABLE 2. 2 COMPARISON DIFFERENCES BETWEEN EACH SOFTWARE DEVELOPMENT TECHNIQUES.....	274

TABLE OF CONTENTS

ACTIVITY 1	24
1.1 PROBLEM DEFINITION STATEMENT	24
1.1.1 <i>Introduction</i>	24
1.1.2 <i>Problem Statement</i>	25
1.1.3 <i>Impact of the Problem</i>	26
1.1.4 <i>Limitations of the Current System</i>	27
1.1.5 <i>Desired Outcomes from the New Solution</i>	29
1.1.6 <i>Risk Areas in Developing the New System</i>	30
1.2 USER REQUIREMENTS	32
1.2.1 <i>Requirements for Customers</i>	32
1.2.2 <i>Requirements for Inventory Handling Clerk</i>	34
1.2.3 <i>Requirements for Production Manager</i>	35
1.2.4 <i>Requirements for Chief Accountant</i>	36
1.2.5 <i>Other User Requirements</i>	37
1.3 SYSTEM REQUIREMENTS.....	38
1.3.1 <i>Functional Requirements</i>	39
1.3.2 <i>Non-Functional Requirements</i>	40
1.4 RISK IDENTIFICATION	42
1.4.1 <i>Types of risks to face</i>	42
1.4.2 <i>Risk Report Forms</i>	43
1.5 SOFTWARE DESIGN DOCUMENT (SDD).....	55
1.5.1 <i>Introduction</i>	55
1.5.2 <i>System Overview</i>	57
1.5.3 <i>User Requirements for Customers</i>	60
1.5.4 <i>User Requirements for Inventory Handling Clerk</i>	61
1.5.5 <i>User Requirements for Production Manager</i>	62
1.5.6 <i>User Requirements for Chief Accountant</i>	63
1.5.7 <i>Other User Requirements</i>	64
1.5.8 <i>System Requirements</i>	66
1.5.9 <i>Existing System Analysis</i>	68
1.5.10 <i>Proposed System Analysis</i>	69
1.5.11 <i>Feasibility Analysis</i>	70
1.5.12 <i>Operational Feasibility Analysis</i>	72
1.5.13 <i>Economic Feasibility Analysis</i>	73
1.5.14 <i>Legal Feasibility Analysis</i>	75
1.5.15 <i>Schedule Feasibility Analysis</i>	76
1.5.16 <i>System Development Methodology in Use</i>	78
1.5.17 <i>System Design</i>	80
1.5.18 <i>System Design – ERD Diagram</i>	82
1.5.19 <i>System Design – Class Diagram</i>	84
1.5.20 <i>System Design – Use Case Diagram</i>	87
1.5.21 <i>System Design – Sequence Diagrams</i>	90
1.5.22 <i>System Design – User Interfaces Design</i>	98
1.5.23 <i>System Design – Logical Database Diagram Design</i>	113
1.5.24 <i>Coding Standards and Conventions</i>	114

1.5.25 Coding Implementation Strategy	116
1.5.26 System Code Implementation	118
1.5.27 Test Plan.....	133
1.5.28 Test Cases.....	137
1.5.29 Test Schedule	151
1.5.30 System Deployment Strategy	153
1.5.31 System Maintenance and Support Plan	154
1.5.32 Risk Identification.....	157
1.5.33 Risk Report Forms	162
1.5.34 Risk Mitigation Plan	174
1.5.35 Conclusion	185
1.6 JUSTIFICATION OF SELECTED SOLUTIONS.....	187
1.6.1 Agile Development Methodology - Scrum:.....	187
1.6.2 UML for System Design:	189
1.6.3 Relational Database:.....	191
1.6.4 Python and Django:.....	193
1.6.5 Conclusion	195
ACTIVITY 2	196
2.1 SOFTWARE DEVELOPMENT TOOLS.....	196
2.1.1 Introduction.....	196
2.2 SOFTWARE DEVELOPMENT TOOLS.....	197
2.2.1 Software Development Tools: Diagram Tools	197
2.2.2 Software Development Tools: Project Management Tools.....	201
2.2.3 Software Development Tools: Programming Tools.....	204
2.2.4 Software Development Tools: Web Development Tools	208
2.2.5 Software Development Tools: Configuration Management Tools	212
2.2.6 Software Development Tools: Quality Assurance Tools	216
2.2.7 Software Development Tools: Documentation Tools	219
2.2.8 Software Development Tools: Design Tools	223
2.2.9 Software Development Tools: Maintenance Tools.....	227
2.2.10 Software Development Tools: Analysis Tools.....	230
2.2.11 Software Development Tools: Process Modelling Tools.....	234
2.2.12 Software Development Tools: Prototyping Tools	237
2.2.13 Software Development Tools: Change Control Tools	241
2.2.14 Comparison differences between each Software Development Tools	244
2.3 SOFTWARE DEVELOPMENT TECHNIQUES.....	246
2.3.1 Types of software development Techniques	246
2.3.2 Software Development Techniques: Waterfall Model	248
2.3.3 Software Development Techniques: Prototype Methodology	250
2.3.4 Software Development Techniques: Spiral Model	252
2.3.5 Software Development Techniques: Agile Methodology	254
2.3.6 Software Development Techniques: Scrum.....	256
2.3.7 Software Development Techniques: Extreme Programming (XP).....	258
2.3.8 Software Development Techniques: Lean Software Development	260
2.3.9 Software Development Techniques: Rapid Application Development (RAD)	262
2.3.10 Software Development Techniques: Joint Application Development (JAD)	264
2.3.11 Software Development Techniques: Dynamic Systems Development Method (DSDM).....	266
2.3.12 Software Development Techniques: Feature Driven Development (FDD).....	268
2.3.13 Software Development Techniques: Test-Driven Development (TDD).....	270
2.3.14 Software Development Techniques: Behavior Driven Development (BDD).....	272

2.3.15 Comparison differences between each Software Development Techniques.....	274
2.3.16 Chosen Development Technique for BAUHINIA Project	278
2.4 CONCLUSION.....	282
ACTIVITY 3	284
3.1 BAUHINIA PROJECT BUSINESS APPLICATION REVIEW	284
3.1.1 Presentation.....	284
3.2 PEER REVIEW	309
3.2.1 Introduction.....	309
3.2.2 Execution of Peer-Review Process	314
3.2.3 Detailed Feedback from Peer Review.....	315
3.2.4 Analysis of Peer Review Feedback.....	317
3.2.5 Incorporation of Feedback into Future Development Cycles.....	319
3.2.6 Conclusion	321
3.3 BUSINESS APPLICATION WITH SUPPORT DOCUMENTATION.....	323
3.3.1 Introduction.....	323
3.3.2 Recap of Software Design Document.....	325
3.3.3 Recap of User Requirements	327
3.3.4 Recap of System Requirements	328
3.3.5 Recap of System Design	330
3.3.6 Selection of Tools, Techniques, and Methodologies.....	349
3.3.7 Development of the Business Application	351
3.4 SUPPORT DOCUMENTATION FOR THE BUSINESS APPLICATION.....	353
3.4.1 User Manual.....	354
3.4.2 Technical Documentation	357
3.4.3 Conclusion	363
3.5 ASSESSMENT OF NEW IDEAS AND POSSIBLE IMPROVEMENTS	365
3.5.1 Mobile Application	365
3.5.2 Integration with Other Systems	367
3.5.3 Advanced Analytics	369
3.5.4 AI and Machine Learning	371
3.5.5 User Customization:	373
ACTIVITY 4	375
4.1 CRITICAL REVIEW INTRODUCTION.....	375
4.1.1 A. Brief Overview of the Inventory Control Application	375
4.1.2 B. Summary of Problem Definition Statement and Initial Requirements	376
4.2 CRITICAL REVIEW OF THE DESIGN STAGE	379
4.2.1 A. Recap of the Design Process - Analysis of the Design Tools Used	379
4.2.2 A. Recap of the Design Process - Effectiveness of the Design created by Designing Tools in Solving the Problem Statement	381
4.2.3 A. Recap of the Design Process - Suitability of the Design created by Designing Tools in Meeting Initial Requirements	383
4.2.4 B. Evaluation of the Design Decisions - Successes and Strengths in the Design Stage created by Designing Tools	385
4.2.5 B. Evaluation of the Design Decisions - Challenges and Areas for Improvement in the Design Stage created by Designing Tools	387
4.3 CRITICAL REVIEW OF THE DEVELOPMENT STAGE	389
4.3.1 A. Recap of the Development Process - Analysis of the Development Tools and Techniques Used	389

4.3.2 A. Recap of the Development Process - Effectiveness of the Development Process in Implementing the Design	391
4.3.3 A. Recap of the Development Process - Suitability of the Developed Application in Meeting Initial Requirements	393
4.3.4 B. Evaluation of the Development Decisions - Successes and Strengths in the Development Stage	395
4.3.5 B. Evaluation of the Development Decisions - Challenges and Areas for Improvement in the Development Stage	397
4.4 CRITICAL REVIEW OF THE TESTING STAGE	399
4.4.1 A. Recap of the Testing Process - Analysis of the Testing Tools and Techniques Used	399
4.4.2 A. Recap of the Testing Process - Effectiveness of the Testing in Ensuring the Quality of the Application	401
4.4.3 A. Recap of the Testing Process - Suitability of the Testing Process in Validating the Initial Requirements	403
4.4.4 B. Evaluation of the Testing Decisions - Successes and Strengths in the Testing Stage	405
4.4.5 B. Evaluation of the Testing Decisions - Challenges and Areas for Improvement in the Testing Stage	407
4.4.6 C. Evaluation of the Test Plan and Test Cases - Test Plan Assessment.....	409
4.4.7 C. Evaluation of the Test Plan and Test Cases - Test Cases Assessment	411
4.5 CRITICAL REVIEW OF THE BUSINESS APPLICATION	413
4.5.1 A. Evaluation of the Business Application Performance against the Problem Definition Statement	413
4.5.2 A. Evaluation of the Business Application Performance against the Initial Requirements	415
4.5.3 B. Evaluation of the Business Application Strengths and Weaknesses	418
4.6 RECAP OF INITIAL RISKS IDENTIFIED	420
4.6.1 A. Brief Summary of the Initial Risks	420
4.6.2 B. Evaluation of How Risks Were Addressed or Materialized During Development	421
4.6.3 C. Evaluation of Risk Identification Effectiveness	423
4.6.4 D. Analysis of Achievements and Strengths in Risk Identification and Mitigation	424
4.7 OPPORTUNITIES FOR IMPROVEMENT.....	426
4.7.1 A. Identified Areas for Enhancement	426
4.7.2 B. Justification for Each Suggested Improvement	429
4.8 SUGGESTIONS FOR FURTHER DEVELOPMENT	432
4.8.1 A. Proposed Additional Features or Changes	432
4.8.2 B. Justification for Each Suggested Development	434
4.9 CONCLUSION OF CRITICAL REVIEW	436
4.9.1 A. Summary of the Critical Review and Evaluations.....	436
4.9.2 B. Reflective Discussion on the Learning Experience and Overall Development Process	439
CONCLUSION	441
REFERENCES.....	442

Activity 1

1.1 Problem Definition Statement

The problem definition statement should clearly state the problem that BAUHINIA is facing, and why it needs a new system. In this case, the problem could be described as: "BAUHINIA is experiencing difficulties managing orders and inventory due to its increased popularity and order volume through social media channels. The current process is manual and time-consuming, leading to errors and inefficiencies."

1.1.1 Introduction

BAUHINIA is a well-known clothing brand in Sri Lanka that has been offering its clients a selection of premium, fashionable clothing since 2018. The company takes pride in its distinctive designs and top-notch customer service, which has contributed to its rapid growth and popularity throughout time. BAUHINIA has made use of the strength of social media sites, mainly Facebook and Instagram, to conduct its sales operations and reach a larger consumer base. Currently, customers can place orders for their desired products by messaging the brand directly and requesting the item code, size, and quantity they need.

The BAUHINIA brand has been facing considerable difficulties as it expands, despite the fact that this method of order processing has largely been successful for the company in its early phases. The complexity of manually managing orders and inventories has increased with the rise in popularity and order volume. This is mainly because physically checking inventories, confirming orders, setting up deliveries, and managing payments for orders is labor-intensive. As a result, this procedure is not only time-consuming and expensive but also leaves space for human mistake, which can result in mismanagement and inefficiencies that may be detrimental to BAUHINIA's reputation and financial health.

The managing director of BAUHINIA has strategically chosen to spend money on a technical solution that would automate the brand's workflow and improve its order and inventory management procedures in response to these difficulties. In order to create this new order tracking system employing an Agile development methodology, BAUHINIA has decided to engage with AKL Software, a software development consultancy. This new online solution is expected to be a considerable upgrade over the current system, with a variety of functions that cater to the demands of both BAUHINIA's staff and its consumers.

The parts that follow will go deeper into the specifics of the difficulty that BAUHINIA is facing, the impact of this problem on the business, the constraints of the current system, the intended outcomes from the new solution, and the goals that the new system should achieve.

1.1.2 Problem Statement

BAUHINIA is facing a major difficulty due to the inefficiencies of their present order processing and inventory management processes. The quantity of orders coming in through their social media channels has increased significantly as the business has grown and gained popularity. The existing order-handling system is manual, requiring individual attention to each customer request.

Order processing includes individual staff members accepting the customer's request, checking the availability of items, confirming the order with the customer, and finally arranging for the item's delivery. This procedure is time-consuming, labor-intensive, and error-prone, which can result in inaccurate order processing, delays, or a lack of communication with the client.

At the same time, inventory management has grown more and more challenging. Manual inventory tracking not only takes a long time, but it also does not provide real-time information about items that are available. Customers may place orders for items that are currently out of stock, resulting in dissatisfaction among consumers and potential damage to BAUHINIA's reputation.

Furthermore, as BAUHINIA has grown, manual handling of these processes has proven unscalable. As the brand grows, the current system cannot handle the growing volume of orders and inventories, posing a substantial barrier to BAUHINIA's future growth and operational efficiency.

The main problem with BAUHINIA is essentially its manual order processing and inventory management system, which is not designed to handle the brand's growth trajectory and associated increase in order volume. This inefficiency is having a negative influence on the company's operations, customer satisfaction, and overall scalability. BAUHINIA must therefore shift to an automated, efficient, and reliable system to streamline its order processing and inventory management.

1.1.3 Impact of the Problem

BAUHINIA's existing system problem has a wide-ranging and complicated impact on various parts of its operations, customer happiness, and overall growth potential.

- To begin with, BAUHINIA's operating efficiency is severely limited by the manual nature of order processing and inventory management.
- The time and resources required to manually handle each order request, confirm it, check inventory, and arrange for delivery are significant.
- This labor-intensive approach is not only expensive, but it also inhibits BAUHINIA's employees from focusing on other critical areas of the business that may boost the brand's growth and profitability.
- Furthermore, there is a lot of possibility for human mistake in the current method.
- Negative customer experiences can result from errors in order processing, such as miscommunication of purchase details, wrong item shipping, or delivery delays.
- The same applies to inventory management. A lack of real-time inventory data might lead to accepting orders for out-of-stock items, further damaging the brand's reputation.

- In terms of customer satisfaction, the absence of a seamless online ordering system could be a disadvantage for BAUHINIA. In the age of digital commerce, consumers have come to expect easy, intuitive, and reliable online shopping experiences.
- The existing method, which requires customers to submit purchases via social media messages, may fall short of these expectations, resulting in lost sales and decreased client loyalty.

From an internal perspective, BAUHINIA's staff also face challenges due to the current system. Manually managing orders and inventories can be difficult and time-consuming. Automating these tasks might significantly decrease their workload, boost job satisfaction, and allow them to better serve their clients.

From an internal perspective, BAUHINIA's staff also face challenges due to the current system. Manually managing orders and inventories can be difficult and time-consuming. Automating these tasks might significantly decrease their workload, boost job satisfaction, and allow them to better serve their clients.

In conclusion, the problem with BAUHINIA's current order processing and inventory management system has wide-ranging impacts, affecting the brand's operational efficiency, customer satisfaction, scalability, and staff motivation. As a result, it is critical for BAUHINIA to take care of this issue and move to an automated system capable of supporting its expansion and improving its operations.

1.1.4 Limitations of the Current System

BAUHINIA's current manual order processing and inventory management system has several significant limitations that hinder its ability to function efficiently and scale its operations.

- Inefficiency: The existing system is extremely inefficient. Receiving and reviewing order requests, checking item availability, confirming orders with customers, and organizing deliveries are all manual tasks in order processing. This method is time-

consuming and labor-intensive, resulting in an inefficient use of BAUHINIA's resources.

- Error-prone: Since the system is manual, it is prone to human error. Order processing errors can result in incorrect items being shipped, improper quantities being sent, or delivery delays. Similarly, manual inventory tracking can result in mistakes in available stock, resulting in orders being accepted for out-of-stock items. These mistakes can cause client displeasure and harm the brand's reputation.
- Lack of Real-Time Inventory Data: The existing system does not give real-time inventory data. This lack of data makes effective stock management difficult and can lead to overstocking or understocking difficulties. It also makes it difficult to offer clients with precise information regarding item availability.
- Poor Scalability: As BAUHINIA expands, the limitations of the current system become more and more apparent. The manual methods are not scalable to meet a higher volume of orders, limiting BAUHINIA's growth potential. It also makes expanding into new markets or channels challenging for the company.
- Limited Customer Experience: The current order method, which requires customers to place orders via social media messages, may fall short of today's digital-savvy consumers' expectations. The absence of an efficient online ordering system can have a negative influence on the customer experience and hinder BAUHINIA's capacity to compete in a world that is increasingly digital.
- Absence of Automated Reporting: The current system does not allow for the generation of automated reports, making it difficult for BAUHINIA to monitor its performance, track its sales, and make well-versed business decisions.

These limitations highlight the urgent need for BAUHINIA to invest in an automated solution that can streamline its order processing and inventory management, provide real-time inventory data, improve scalability, enhance the customer experience, and enable automated reporting.

1.1.5 Desired Outcomes from the New Solution

Given the limitations of BAUHINIA's current system, the company is looking for a new online solution that can successfully automate order processing and inventory management, thereby improving both operational efficiency and customer experience. The following are the desired outcomes of the new system:

- Improved operational efficiency: The new system is expected to streamline BAUHINIA's operations, removing the need for manual order processing and inventory management. Automation should reduce labor-intensive tasks, saving time and resources.
- Reduced Errors: By automating order processing and inventory tracking, the new system should reduce the risk of human error, ensuring that orders are handled correctly and inventory levels are accurately monitored.
- Real-Time Inventory Tracking: The new solution should give real-time data on inventory levels, assisting in the prevention of overstocking and understocking issues, as well as guaranteeing that customers receive accurate information on item availability.
- Enhanced Scalability: As BAUHINIA expands, the system should be able to manage a higher amount of orders. The solution should be scalable, allowing BAUHINIA to grow without having to deal with the challenges of manual order processing and inventory management.
- Improved Customer Experience: The new online system should give customers with a smooth and simple user experience, allowing them to simply explore products, check availability, add items to their cart, and checkout. This enhancement should match customers' online purchasing expectations, potentially leading to higher sales and customer loyalty.
- Automated Reporting: The system should allow for the automated generation of daily and monthly reports, which will aid in performance monitoring, sales tracking, and informed decision-making.
- Secure User Registration and Sign-in: The system should allow customers and workers to register and sign in securely, maintaining the security of user data.

- Flexible Access Levels: The system should provide several levels of access, such as "report only," "update only," "update and delete," and "complete system access," to enable staff to do their responsibilities effectively and safely.

The system that is being suggested will be a considerable improvement over the current one, delivering a more efficient, accurate, and scalable solution for order processing and inventory management at BAUHINIA. Furthermore, it will considerably improve BAUHINIA's customers' purchasing experience and help the brand compete more successfully in the digital economy.

1.1.6 Risk Areas in Developing the New System

While the benefits of a new system for BAUHINIA are clear, there are several risks associated with its development that need to be considered and mitigated:

- Technical Complexity: The development of an online solution that can handle BAUHINIA's order processing and inventory management, provide real-time inventory data, enable user registration and sign-in, and generate automated reports is technically complex. There's a risk that technical challenges or limitations might arise during the development process that could delay the project or increase costs.
- Data Migration: Moving existing customer and inventory data from the current manual system to the new automated system will be required. There is a possibility that some data will be lost, corrupted, or moved incorrectly during this process, affecting the functionality of the new system and potentially disrupting BAUHINIA's operations.
- Security of the system: The new system will manage sensitive client data such as names, email addresses, delivery addresses, and phone numbers. The system may be exposed to cyberattacks, which might compromise this data and harm BAUHINIA's reputation.
- User Acceptance: While the new system is expected to improve operational efficiency and customer experience, there's a risk that BAUHINIA's staff or

customers might resist the change. If they find the new system difficult to use or does not satisfy their needs, the effectiveness of the solution may be limited, affecting BAUHINIA's operations or sales.

- Cost Overruns: Due to the system's complexity, there is a chance that the project will go over budget due to unforeseen technological issues, longer timescales, or higher resource requirements.
- Dependency on Vendor: BAUHINIA intends to hire a software development consultant to help design the new system. There's a risk that the consultant might not deliver the project on time, to the expected quality, or within the agreed budget. This could cause the new system's implementation to be delayed, thereby affecting BAUHINIA's operations.

Given these risks, BAUHINIA and its chosen software development consultant must take aggressive measures to mitigate them. This could involve adopting a rigorous project management approach, implementing robust data migration and security protocols, providing comprehensive training to staff and customers, and closely managing the relationship with the software development consultant.

1.2 User Requirements

The special demands or expectations of the software's users are reflected in the user requirements. Such as what functions users must be able to carry out. In a User Requirements Document (URD), narrative material is typically used to describe user requirements. The user typically approves user requirements, which are the main source of information for formulating system requirements.

Finding out what the user truly wants a piece of software to do is a crucial and challenging phase in its development. This is due to the fact that the user frequently struggles to fully express their requirements and desires, and that the information they supply may also be unreliable, imprecise, or inconsistent.

This will involve understanding and documenting what the different types of users need from the system. We have different types of users like Customers, Inventory handling Clerk, Production Manager, and Chief Accountant. We will need to map out the tasks they need to complete, their information needs, and any particular preferences or constraints they have. For example, customers need to be able to register, log in, browse products, add them to a cart, and check out.

1.2.1 Requirements for Customers

The new system should provide BAUHINIA customers with a user-friendly and secure online buying experience that matches their expectations and is consistent with current eCommerce practices.

- User Sign-In and Registration: Customers should be able to register on the BAUHINIA platform by entering their name, email address, delivery address, password, and two active phone numbers. The registration process should be simple and straightforward. Customers should be able to check in to their accounts using

their email address and password after registering. The system should ensure user data security and password security.

- Browse and Search for Products: Customers should be able to quickly navigate BAUHINIA's product catalog. The system should provide effective search and filtering features to assist customers in quickly and efficiently finding specific goods or product categories.
- Check Product Availability: Customers should be able to see a product's availability status in real-time when viewing it. If a product is out of stock, the system should notify the customer and possibly offer the option to be notified when it is back in stock.
- Add Items to Cart: Customers should be able to add items to an online shopping cart, selecting the quantity they want for each item. Before going through with the checkout, they ought to be able to review their shopping cart, change the quantities, or delete items.
- Checkout Process: Customers should get a summary of their order, including the total amount to be paid, during the checkout process. They should be able to confirm their billing information as well as select their chosen payment method. As BAUHINIA currently uses a cash-on-delivery system, this option should be clearly presented. The system should then confirm the successful placement of the order, providing the customer with an order reference number for tracking purposes.
- View and change Personal Information: Customers should be able to view and change their personal information in their account settings, including their delivery address and contact numbers. To protect client data, the system should ensure that any such updates are done securely.

The customer-facing aspects of the new system should focus on enhancing the user experience, promoting user engagement, and facilitating secure and efficient online shopping.

1.2.2 Requirements for Inventory Handling Clerk

The inventory handling clerk plays a key role in BAUHINIA's operations, and the new system should provide a robust set of tools to facilitate the efficient management of inventory.

- User Registration and Sign-In: The registration and sign-in process for inventory handling clerks should be secure, just like it is for consumers. They should use their professional email address and a secure password to sign up. The system should offer different functionalities in accordance with the differences between customer and staff accounts.
- Add New Items to Inventory: The inventory handling clerk should have the ability to add new items to the inventory. This process should include adding detailed product information, such as item code, product description, size variations, price, and initial quantity. The system should validate these inputs to ensure accuracy and consistency in the inventory data.
- Update Existing Item Details: The clerk should be able to change existing inventory item data. This could include modifying the item's description, pricing, or available quantities, as well as terminating it. Any updates should be reflected in real time in the customer-accessible product catalog.
- Access to Real-Time Inventory Data: The system should provide the clerk with real-time data on inventory levels. This should help the clerk maintain optimal stock levels, restocking items when necessary, and preventing overstock or understock situations. The system could provide alerts when inventory levels for certain items fall below a predefined threshold.
- View and Update Personal Information: Just like customers, inventory handling clerks should be able to view and update their personal information securely within the system.

By meeting these requirements, the new system will enable the inventory handling clerk to manage BAUHINIA's inventory more effectively and efficiently, contributing to smoother operations and better customer service.

1.2.3 Requirements for Production Manager

The Production Manager at BAUHINIA plays a crucial role in ensuring the smooth flow of operations, from inventory management to order fulfillment. The new system should be equipped with features that can significantly streamline these processes and facilitate decision-making.

- **User Registration and Sign-In:** The Production Manager should be able to register and sign in to the system in a secure manner. Individual credentials should be unique, ensuring safe access to system functions relevant to their role.
- **Generate Daily Orders Report:** A crucial part of the Production Manager's role is the overview and management of the orders placed by the customers. The system should allow the Production Manager to generate daily reports outlining the orders placed, their status, and other related information. This data can help track sales performance, identify any issues, and ensure that the order fulfillment process is running smoothly.
- **Generate Daily Product Availability Report:** Another critical function is the ability to generate product availability reports. This feature would make it possible for the Production Manager to assess current stock levels, spot products that might need to be ordered again soon, and make sure that high-demand items are always available. To ensure that these reports are accurate, the system should deliver real-time data.
- **Access to Real-Time Inventory Data:** In addition to generating reports, the system should provide the Production Manager with real-time access to inventory data. This allows for an immediate overview of the available stock and assists in making quick decisions regarding inventory management.

- View and Update Personal Information: Lastly, the Production Manager should be able to securely view and update their personal information within the system.

By fulfilling these requirements, the new system would provide valuable support to the Production Manager, promoting effective oversight and management of BAUHINIA's operations.

1.2.4 Requirements for Chief Accountant

As the person responsible for the financial overview of BAUHINIA, the Chief Accountant needs specific tools and functionalities to effectively track, analyze, and report the company's financial performance.

- User Registration and Sign-In: The Chief Accountant should have secure access to the system, ensuring the security of sensitive financial data. They, like all other users, should register and sign in with unique credentials.
- Generate Monthly Income Report: An essential part of the Chief Accountant's role is the ability to generate detailed monthly income reports. The system should automatically gather and show important information, such as total sales, returns, discounts, and cash on delivery collected, in a format that enables for simple analysis and reporting. Because these reports have a direct impact on the company's financial goals and decisions, the system should assure data integrity and accuracy.
- View and Update Personal Information: The Chief Accountant, like other users, should be able to securely view and update their personal information within the system.

By offering these functionalities, the new system may greatly simplify the Chief Accountant's workflow and guarantee that they have access to the data they need to maintain BAUHINIA's financial health. This also enhances overall business transparency.

because timely and accurate financial reporting are essential for strategic planning and top-level decision-making inside the organization.

1.2.5 Other User Requirements

In addition to the specific requirements of the various user roles, there are several general requirements that all users, regardless of their role, will need from the new system. These primarily revolve around usability, security, performance, and scalability.

- **User-Friendly Interface:** Whether a customer is looking for products or an employee is managing inventory, all users should find the system simple to access and use. The user interface should be simple to use minimizing the learning curve for new users and facilitating efficient use. Appearance should be considered as well, to ensure that the system is visually appealing and matches with BAUHINIA's brand image.
- **Secure Access and Data Protection:** Since the system handles sensitive user data, such as personal information and order details, comprehensive security measures are required. These could include data encryption, safe communication protocols, and frequent security assessments. User access should be controlled by unique credentials, and the system should log user activity for audit purposes.
- **Reliable Performance and Availability:** Users should be able to access the system whenever they need to, and it should respond quickly to their interactions. This requires reliable hosting, efficient code, and regular maintenance to identify and resolve any performance issues. Downtime should be kept to a minimum to avoid disrupting BAUHINIA's activities and ensuring a positive customer experience.
- **Scalability:** As BAUHINIA grows, the system should be able to scale to handle more users and larger volumes of data. This requires a scalable architecture and planning for potential upgrades to system resources.
- **Support and maintenance:** Ongoing support and maintenance are critical to ensuring the system's continued operation and usefulness. This includes debugging and resolving difficulties, installing and upgrading software, and giving user assistance.

By meeting these general requirements, the new system can ensure a positive experience for all users while also maintaining the security, reliability, and scalability necessary for BAUHINIA's ongoing success.

1.3 System Requirements

The building blocks that developers utilize to construct the system are called system requirements. These statements outline what the system "must" perform. These requirements, which the customer provides in order to be satisfied, are both wide and precise. The customer should express exactly what they want and how they want it in the statement. Functional requirements and Non-Functional requirements are the two categories of system requirements.

A Functional Requirement outlines what a user requires in order to complete their task. An example of a functional requirement would be for a system to be able to enter and print cost estimates. All other requirements not covered by the functional requirements are described as non-functional requirements. The system architecture includes specifics about how to implement non-functional requirements, such as accessibility, speed, and performance.

1.3.1 Functional Requirements

These are specific functionalities or services the system should provide.

- User Registration: The system should provide a registration feature for both customers and staff members, allowing them to create an account by providing necessary details such as name, email address, delivery address, password, and telephone numbers. For staff members, their professional role should also be specified.
- User Authentication: The system should include a secure sign-in process that requires users to enter their registered email address and password. This will help to protect user accounts from unauthorized access.
- Secure Access Levels: To safeguard system functionality, the system should use role-based access control. A user should be provided appropriate permissions based on their role (customer, inventory handling clerk, production manager, or chief accountant), such as read-only, update-only, update-and-delete, or entire system access.
- Product Browsing: Customers should be able to explore a product catalog that is organized by category for ease of browsing. Each product listing should include information such as the item code, description, available sizes, price, and state of availability.
- Shopping Cart Management: Customers should be able to add items to a virtual shopping cart, see the contents of the cart at any moment, change quantities, and delete items as needed.
- Checkout and Order Placement: Once customers have finished adding items to their cart, they should be able to proceed to checkout. The system should calculate the total amount, allow customers to confirm their billing details, and place the order.
- Inventory Management: The inventory handling clerk should be able to add new items to the inventory, edit the details of existing products, and examine real-time inventory data using the system.

- Order Tracking: After an order is placed, the system should provide customers with updates on the status of their order until it is delivered.
- Reporting: The system should facilitate the creation of various reports. The production manager should be able to generate daily reports of orders and product availability, and the chief accountant should be able to create a monthly income report.
- Personal Information Management: All system users should be able to securely see and update their personal information. This includes contact information and password changes. To preserve the privacy and security of user data, the system should adhere to data protection guidelines.

1.3.2 Non-Functional Requirements

These define the system's properties or characteristics and how the system should behave.

- Performance: The system should be built for speed and responsiveness, allowing users to navigate, browse products, and carry out tasks without encountering any delays. This requires efficient algorithms, lightweight design, and appropriate use of system resources.
- Availability: The system should be reliably available, minimizing downtime to avoid disrupting BAUHINIA's business operations or inconveniencing customers. This may involve using reliable hosting services, implementing failover mechanisms, and performing regular maintenance.
- Scalability: As BAUHINIA grows, the system must be able to handle an increasing number of users and higher volumes of data. This may involve using a scalable architecture, planning for server upgrades, and optimizing code for efficiency.
- Usability: The system should be user-friendly, allowing customers and staff to navigate and use its features with ease. This involves having an easy-to-use layout, clear instructions, and a visually appealing design.

- Security: To secure user data and prevent illegal access, the system should adopt strong security measures. This may involve data encryption, secure transmission protocols, intrusion detection systems, and regular security audits.
- Maintainability: The system should be easy to maintain and upgrade, allowing for bug fixes, functionality enhancements, and adaptation to changing business needs. This requires well-documented and modular code, a strong development workflow, and adherence to software development best practices.
- Compatibility: The system should work properly across all devices and browsers, ensuring that all users, regardless of device or browser preference, can utilize it efficiently. This necessitates responsive design and thorough cross-browser testing.
- Interoperability: Other software systems, such as a customer relationship management system or an accounting system, may require integration with the system. It should be built with interoperability in mind, making use of common data formats and communication protocols.
- Compliance: The system should comply with relevant regulations and standards, such as those related to data protection, e-commerce, and accessibility. This involves both technical design choices and business processes.
- Disaster Recovery: In the event of a system failure or data loss, the system should have backup and recovery options. This could include frequent data backups, a disaster recovery plan, and data recovery tools.

1.4 Risk Identification

1.4.1 Types of risks to face

- Technical Risks: These risks relate to the technical aspects of software development, including the use of new technologies or tools that the development team might not be familiar with, or technical issues that might arise during the development process. For instance, there could be issues related to data migration from the old system, or the integration of the new system with existing systems.
- Project Management Risks: These risks relate to the coordination, planning, and management of the software development process. For instance, there could be risks of delays due to poor planning, miscommunication among the team, or scope creep where new features are continuously added to the project, causing it to deviate from the original plan.
- Financial Risks: These risks relate to the financial aspects of the project, such as budget overruns due to unforeseen costs, or failure to achieve the desired return on investment if the system does not meet its objectives or is not adopted by the users as expected.
- Operational Risks: These risks involve potential challenges that could arise during the system's operational life. For instance, there could be risks related to system downtime, performance issues, or difficulties in maintaining and upgrading the system.
- Legal and Compliance Risks: These risks refer to potential legal issues or non-compliance with regulations, such as data protection laws. For instance, if the system does not adequately protect customer data, BAUHINIA could face legal penalties, as well as damage to its reputation.
- Security Risks: These risks refer to potential vulnerabilities and threats that could compromise the system's security. For instance, there could be risks of data breaches, unauthorized access, or other forms of cyberattacks.
- Usability Risks: These risks involve the potential for the system to be difficult to use or not meet user expectations. For instance, if the system's user interface is not

intuitive, or if it does not provide the expected functionalities, users might not adopt the system, leading to a failure in achieving the project's objectives.

- Vendor and Third-party Risks: These risks involve dependencies on external parties, such as software vendors or service providers. For instance, there could be risks related to the quality of third-party services, or delays or disruptions due to issues on the vendor's side.
- Organizational Change Risks: These risks pertain to the potential challenges associated with introducing a new system within the organization. For instance, there could be resistance from employees who are used to the old system, or difficulties in training employees to use the new system.
- Scalability Risks: These risks relate to the system's ability to scale and handle increased user load or data volume. For instance, if BAUHINIA's business grows rapidly, the system might not be able to handle the increased demand, leading to performance issues or system failures.

1.4.2 Risk Report Forms

Table 1. 1 RISK REPORT FORM – For Technical Risks

RISK REPORT FORM – For Technical Risks	
Date	9 th June 2023
Risk description	BAUHINIA may face significant technical difficulties during the transition from a manual to a computerized system. This includes potential challenges in data migration, system integration, and managing the learning curve for both staff and customers.
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not strictly date driven. However, it may arise during the transition phase when the new system is being implemented and integrated with existing processes.

Deliverables impacted	The risk could potentially impact multiple deliverables, including system deployment, staff training, customer adaptation, and even the regular business operations of BAUHINIA during the transition period.	
Initial assessment	Impact : High	Likelihood : Moderate
Suggested risk response	<p>BAUHINIA could engage an experienced IT consultant to oversee the transition process and provide necessary technical guidance.</p> <p>Comprehensive training programs should be implemented to familiarize staff with the new system. User manuals and help guides can be developed for reference.</p> <p>For customers, a robust support system can be established to address their queries and concerns about the new system. This includes a FAQ section on the website, customer service hotlines, and instructional videos.</p> <p>Regular backups of business data should be maintained to prevent loss during the transition. A rollback plan should also be in place in case significant issues arise during the system switch.</p> <p>Gradual rollout of the system can be considered, starting with non-critical operations to ensure smooth transition and minimal disruptions to business.</p>	

Table 1. 2 RISK REPORT FORM – For Project Management Risks

RISK REPORT FORM – For Project Management Risks		
Date	9 th June 2023	
Risk description	BAUHINIA's may encounter management-related risks such as poor coordination among team members, miscommunication, or scope creep, leading to delays in project completion.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	Yes, this risk is date-driven. Poor project management could lead to delays, which would impact the predetermined launch date of the new system.	
Deliverables impacted	This risk could potentially impact all project deliverables due to potential delays, miscommunication, and lack of coordination. It could directly impact the development, testing, and deployment of the new system.	
Initial assessment	Impact : Significant	Likelihood : Probable
Suggested risk response	<p>BAUHINIA should consider hiring an experienced project manager who is familiar with managing similar software development projects.</p> <p>Implementing a clear project plan and a strong communication strategy could help to avoid miscommunication and keep all team members aligned.</p> <p>Regular project status meetings should be held to discuss progress, address any issues, and make necessary adjustments to the plan.</p> <p>A clear scope statement should be defined at the beginning of the project, and any changes to the scope should be strictly managed to avoid scope creep.</p> <p>Utilize project management tools for better coordination, timeline management, and communication among team members.</p>	

Table 1. 3 RISK REPORT FORM – For Financial Risks

RISK REPORT FORM – For Financial Risks		
Date	9 th June 2023	
Risk description	BAUHINIA's transition to a digital solution may encounter financial risks. These could arise from unforeseen development costs, overruns on the budget, or inadequate returns on the investment if the system does not meet its objectives or is not as widely adopted as expected.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not strictly date driven. However, it could become significant during any phase of the project if costs exceed the budget or if the expected returns on the investment are not realized post-implementation.	
Deliverables impacted	This risk could potentially affect the entire project as it involves the financial sustainability of the project. Any financial instability could lead to compromises in the quality of the system or delays in the project timeline.	
Initial assessment	Impact : Moderate	Likelihood : Occasional
Suggested risk response	<p>Proper budget planning and allocation should be carried out at the beginning of the project. Regular budget reviews and updates should be made to manage and control costs.</p> <p>For handling potential cost overruns, a contingency fund should be set aside.</p> <p>BAUHINIA should conduct a comprehensive cost-benefit analysis to ensure the project is financially viable.</p> <p>Monitor project progress closely to ensure it aligns with the project budget and timeline.</p> <p>Use of cost management tools can help in effective financial management.</p>	

	Plan for an effective marketing strategy to ensure high adoption rates and good returns on the investment.
--	--

Table 1. 4 RISK REPORT FORM – For Operational Risks

RISK REPORT FORM – For Operational Risks		
Date	9 th June 2023	
Risk description	Operational risks such as system downtime, performance issues, and difficulties in maintaining and upgrading the new system might affect BAUHINIA's transition to a new order tracking system.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Operational risks can arise at any time during the system's lifecycle, including after its successful implementation.	
Deliverables impacted	If operational risks materialize, they could affect the system's performance, user experience, and overall success. In extreme cases, they could even disrupt BAUHINIA's business operations.	
Initial assessment	Impact : Low	Likelihood : Occasional
Suggested risk response	<p>To minimize downtime, BAUHINIA should ensure that the system is thoroughly tested before deployment. Moreover, an IT support team should be on standby to quickly respond to any system issues that may arise.</p> <p>To ensure system performance, regular system maintenance should be scheduled, and any performance issues should be promptly addressed.</p> <p>For maintenance and upgrades, BAUHINIA should consider setting up a contract with the system developer or a third-party service provider. This would ensure that</p>	

	<p>the system stays up-to-date and compatible with future business needs and technologies.</p> <p>BAUHINIA should develop a disaster recovery plan to address potential crises, such as system failures. The plan should outline procedures for data recovery, system restoration, and business continuity.</p>
--	---

Table 1. 5 RISK REPORT FORM – For Legal and Compliance Risks

RISK REPORT FORM – For Legal and Compliance Risks		
Date	10 th June 2023	
Risk description	Legal and compliance risks, such as non-compliance with data protection laws and regulations, can pose a significant risk to the successful completion and operation of BAUHINIA's new order tracking system.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Legal and compliance risks can arise at any time during the system's lifecycle. They can arise from changes in laws and regulations or from inadvertent non-compliance.	
Deliverables impacted	The risk could potentially impact all aspects of the system that involve data handling and user privacy. This includes, but is not limited to, customer registration, product checkout, and report generation.	
Initial assessment	Impact : Catastrophic	Likelihood : Frequent
Suggested risk response	<p>BAUHINIA should conduct a thorough legal review of the proposed system to ensure compliance with all applicable laws and regulations, particularly those related to data protection and privacy.</p> <p>Implement strong data encryption and security measures to protect user data.</p>	

	<p>Regularly update terms of service and privacy policies to reflect changes in law or system practices.</p> <p>Provide necessary training to staff on handling and protecting user data.</p> <p>Consider obtaining professional legal advice, especially when the system undergoes significant changes or when laws and regulations change.</p> <p>Conduct regular audits of system operations and practices to ensure ongoing compliance.</p>
--	---

Table 1. 6 RISK REPORT FORM – For Security Risks

RISK REPORT FORM – For Security Risks		
Date	10 th June 2023	
Risk description	BAUHINIA's new online order tracking system may be susceptible to various security threats such as unauthorized access, data breaches, and cyber attacks. This could compromise customer information and sensitive business data.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Security risks can occur at any time, especially if appropriate security measures are not in place.	
Deliverables impacted	This risk can affect the confidentiality, integrity, and availability of the online order tracking system. It can also damage BAUHINIA's reputation, customer trust, and could have legal implications due to non-compliance with data protection regulations.	
Initial assessment	Impact : Catastrophic	Likelihood : Probable
Suggested risk response	Ensure the application of standard security practices during the software development process, including secure coding practices and regular security audits.	

	<p>Implement robust authentication and authorization mechanisms to prevent unauthorized access.</p> <p>Use secure communication protocols to protect data during transmission.</p> <p>Regularly update and patch the system to address security vulnerabilities.</p> <p>Regularly back up data and ensure it can be restored quickly in case of a data breach.</p> <p>Educate staff about security best practices, such as recognizing phishing attempts, using strong passwords, and reporting suspicious activities.</p>
--	--

Table 1. 7 RISK REPORT FORM – For Usability Risks

RISK REPORT FORM – For Usability Risks		
Date	10 th June 2023	
Risk description	There's a risk that the new system developed for BAUHINIA may not meet the usability expectations of the users, including both the employees and customers. This could result in low adoption rates, inefficiencies, and customer dissatisfaction.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk isn't date driven. However, it could surface anytime during the system's operational phase, particularly when users start interacting with the system.	
Deliverables impacted	This risk could potentially affect a range of deliverables, most notably, user satisfaction, system adoption, operational efficiency, and customer loyalty.	
Initial assessment	Impact : Negligible	Likelihood : Remote
Suggested risk response	Prioritize user-centric design in the development of the system, incorporating principles of user experience (UX) and user interface (UI) design.	

	<p>Conduct usability testing with a diverse group of potential users (including both staff and customers) to gain feedback and make improvements before the system is fully implemented.</p> <p>Establish a feedback mechanism, such as a user survey or a suggestion box, for continuous improvement post-implementation.</p> <p>Develop comprehensive user guides, conduct training sessions for staff, and provide easily accessible customer support to facilitate the transition to the new system.</p> <p>Include options for personalization or customization within the system to cater to individual user preferences. This could enhance user acceptance and satisfaction.</p>
--	--

Table 1. 8 RISK REPORT FORM – For Vendor and Third-party Risks

RISK REPORT FORM – For Vendor and Third-party Risks		
Date	10 th June 2023	
Risk description	BAUHINIA may face potential risks related to reliance on external software vendors or service providers. This could involve potential issues such as unavailability of service, poor quality, or non-compliance with agreed terms and conditions, which could impact the success of the application.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	Yes, this risk could potentially arise during key dates or events such as the system deployment or during system upgrades and maintenance activities. The risk could also become prominent when there is a need for immediate support or service from the vendor.	
Deliverables impacted	The risk could potentially impact the overall system quality, system uptime, system maintenance and support, and even the project timeline if there are significant delays or issues from the vendor side.	
Initial assessment	Impact : Moderate	Likelihood : Remote

Suggested risk response	<p>BAUHINIA should conduct thorough due diligence before selecting vendors or service providers, including checking their past performance, reputation, financial stability, and compliance with industry standards.</p> <p>Clear and comprehensive contracts should be established with vendors, outlining the expected level of service, penalty clauses for non-compliance, and exit clauses.</p> <p>Regular performance review meetings should be conducted with vendors to ensure they are meeting the agreed standards. Any issues should be addressed promptly.</p> <p>BAUHINIA should have a contingency plan in place in case of significant vendor-related issues, such as having alternate vendors identified or being prepared to bring certain activities in-house if necessary.</p>
-------------------------	---

Table 1. 9 RISK REPORT FORM – For Organizational Change Risks

RISK REPORT FORM – For Organizational Change Risks		
Date	10 th June 2023	
Risk description	BAUHINIA may encounter resistance to organizational change when implementing the new online ordering system. Both employees and customers might struggle to adapt to the new processes, impacting the adoption rate and overall effectiveness of the system.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	This risk is not strictly date driven, but it could become more pronounced during the initial rollout and first few months of the new system's operation.	
Deliverables impacted	The risk could potentially affect several deliverables, including staff training, customer satisfaction, user adoption rates, and the efficiency of business operations during the transition period.	
Initial assessment	Impact : Significant	Likelihood : Probable

Suggested risk response	<p>BAUHINIA should plan for comprehensive training and onboarding sessions for staff members to understand and familiarize themselves with the new system. This will include hands-on practice, demos, and user manuals.</p> <p>A transition period during which both the old and new systems run in parallel could be beneficial. This allows users to adjust to the new system gradually without the pressure of immediate full adoption.</p> <p>BAUHINIA should prepare for customer support and assistance in the initial stages. This could include creating a robust FAQ section on the website, chatbot assistance, customer service hotlines, and instructional videos.</p> <p>Gathering feedback from both employees and customers regularly can help identify pain points and make necessary adjustments to the system or its implementation process. This promotes a sense of inclusion in the change process, potentially easing resistance.</p>
-------------------------	--

Table 1. 10 RISK REPORT FORM – For Scalability Risks

RISK REPORT FORM – For Scalability Risks		
Date	10 th June 2023	
Risk description	As BAUHINIA's business continues to grow, the new system may face scalability issues, unable to handle the increased user load, transaction volumes, and data storage requirements.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	This risk is not date driven, but is instead dependent on the rate of BAUHINIA's business growth, which can fluctuate over time.	
Deliverables impacted	This risk could impact system performance, user satisfaction, and the overall operational efficiency of BAUHINIA. It could potentially affect key deliverables such as system responsiveness, transaction processing speed, and data storage and retrieval efficiency.	
Initial assessment	Impact : Low	Likelihood : Probable

Suggested risk response	<p>During the system design phase, consider scalability as a primary requirement. The system architecture should be flexible and adaptable, allowing for scaling up or down as per demand.</p> <p>Regularly monitor system performance and user load. Use these insights to foresee potential scalability issues and take proactive measures.</p> <p>Explore cloud solutions, which offer scalability as a feature. It allows resources to be added or removed as required, thus providing flexibility.</p> <p>Perform stress testing and load testing. This helps identify the system's breaking point and provides insights into how it would behave under peak loads.</p> <p>Consider implementing a microservices architecture, where different functionalities of the system are divided into small, independent services. This allows specific services to be scaled up or down based on demand.</p>	

1.5 Software Design Document (SDD)

1.5.1 Introduction

1.1 Purpose of the Document

The Software Design Document (SDD) contains a detailed architectural overview of BAUHINIA's proposed order tracking system. This document acts as a blueprint, defining the system's functionality, how system components interact, and the activities involved in its development, implementation, and maintenance. It serves as a reference point for many stakeholders, such as developers, project managers, and clients, to guarantee a common understanding of the system and its goals.

1.2 Document Conventions

This document follows the norms for a software design document. It is divided into sections, each of which addresses a different component of the software design process. For ease of reference, each section is numbered, and technical words and terminology are described when they are first introduced.

1.3 Intended Audience and Reading Suggestions

This document's core readership consists of the software development team, project managers, and the client - BAUHINIA. This document can be used by software developers to understand the system's architecture as well as the coding and implementation methodologies. Project managers can use this to monitor project progress and coordinate tasks. As the client, BAUHINIA can use this document to understand how the suggested solution could meet their needs.

1.4 Project Scope

The project scope consists of designing, developing, testing, deploying, and maintaining an online order tracking system for BAUHINIA. This system aims to replace BAUHINIA's current manual order handling process with an automated, efficient, and user-friendly solution. The system will allow customers to register, browse products, place orders, and track their orders. It will also provide inventory management, order reporting, and income reporting capabilities tailored to the various user roles within BAUHINIA.

1.5 References

Any references used for the creation of this document, including technical guides, requirement documents, and existing literature, are included in this section. This provides a resource for further reading and a context for the proposed system's design and development.

1.5.2 System Overview

2.1 System Architecture

A well-known software application architecture is three-layer architecture, which divides applications into three logical and physical computing levels. The presentation layer is made up of the user interface. The application layer, which processes data, and the data layer, which stores and manages the data linked with the application. The main advantage of three-layer architecture is that because each layer runs on its own infrastructure, each layer may be developed simultaneously by a separate development team and updated or scaled as needed without affecting the other layers.

The proposed order tracking system for BAUHINIA will adopt a three-tier architecture, a common structure for modern web applications. This architecture comprises the Presentation Layer, Application Layer, and Database Layer.

Three-Layer Architecture for BAUHINIA proposed system →

Presentation Layer

- Display Products: Show all the products available for purchase.
- Product Search: Allows users to search for products using various parameters like product name, category, size, color, etc.
- Registration: Allow new users to create an account with all necessary details.
- Login: Provide registered users with the ability to log in to their accounts.
- Shopping Cart: Let users add desired products to a shopping cart and modify the quantity of each item.
- Order Placement: Facilitate customers to place orders, providing necessary details such as delivery address and payment method.
- Order Tracking: Display the status of each order placed by the user.

Application Layer

- User Authentication: Authenticate users when they attempt to log in, ensuring only registered users access the system.
- Role-Based Access Control: Differentiate functionalities accessible by different user types (customer, inventory handling clerk, production manager, chief accountant).
- Order Processing: Process all orders placed, from the initiation of an order to the completion of delivery.
- Inventory Management: Manage inventory details, handle the decrease in inventory with each order, and alert when reordering is necessary.
- Report Generation: Generate required reports based on the operations of the system (daily orders, product availability, monthly income).

Database Layer

- User Data Management: Store and manage all user details securely.
- Product Data Management: Maintain the data of all the products available.
- Order Data Management: Keep a record of all orders placed, their details, and status.
- Inventory Data Management: Maintain accurate inventory records, update with every order placement and product addition/removal.
- Report Data Management: Store and manage all generated reports for future reference.

- The **Presentation Layer** will serve as the system's front-end, acting as the interface through which users interact with the program. It will be designed to provide a user-friendly experience, with simple navigation and controls. This layer will show the available products, allow customers to register and login, and allow them to place and track orders.
- The **Application Layer**, also known as the Business Logic Layer, is in charge of handling user requests, administering business rules, and coordinating responses between the Presentation and Database layers. This layer will make user authentication, inventory changes, order processing, and report generation easier.
- The **Database Layer** will serve as the system's back end, storing and managing all customer, order, inventory, and report data. It will ensure the integrity, reliability, and security of the data.

2.2 Major System Components and their Interactions

The system will consist of several major components, including the User Module, Order Module, Inventory Module, and Reporting Module.

- The **User Module** will handle all operations related to the users of the system. It will process customer registration, sign-in, and profile management. For BAUHINIA staff, it will also handle role-based access control.
- The **Order Module** will facilitate the product selection, cart management, and order placement by customers. It will also track the orders, update their status, and allow customers to view their order history.
- The **Inventory Module** will be utilized by the inventory handling clerk to manage the inventory, add new items, and update existing items' details.
- The **Reporting Module** will generate daily and monthly reports on orders, product availability, and income. This will be primarily used by the Production Manager and Chief Accountant.

These modules will interact with each other, sharing data and triggering actions based on the defined business logic. For example, an order placement in the Order Module will

trigger an inventory update in the Inventory Module, and if the inventory falls below a certain level, it could trigger a reordering alert.

BAUHINIA proposed system 4 major modules with its specific functionalities →

User Module**• Customer Sub-module**

- Customer Registration: Customers will be able to create a new account by providing their name, email address, delivery address, password, and two working telephone numbers.
- Customer Login: Customers will be able to sign in to their account using their email address and password.
- Profile Management: Customers will be able to view and update their profile information.

• Staff Sub-module

- Staff Registration: Authorized personnel can create new staff accounts.
- Staff Login: Staff members can sign in to their accounts.
- Role-based Access Control: Staff members will have access to certain functionalities based on their role (e.g., Inventory handling Clerk, Production Manager, Chief Accountant).

Order Module

- Product Browsing: Customers will be able to view all available products.
- Cart Management: Customers can add products to their cart, view the cart, and make modifications if necessary.
- Order Placement: Customers can place orders for the items in their cart.
- Order Tracking: Customers can track the status of their orders.

Inventory Module

- Inventory Viewing: The clerk can view all items in the inventory along with their details.
- Adding New Items: The clerk can add new items to the inventory.
- Updating Existing Items: The clerk can update details of existing items, such as their quantity, price, and description.

Reporting Module

- Daily Order Report: The Production Manager can generate a report of all orders requested on a given day.
- Daily Product Availability Report: The Production Manager can generate a report of product availability.
- Monthly Income Report: The Chief Accountant can generate a report showing the income for a particular month.

1.5.3 User Requirements for Customers

The new system should provide BAUHINIA customers with a user-friendly and secure online buying experience that matches their expectations and is consistent with current eCommerce practices.

- **User Sign-In and Registration:** Customers should be able to register on the BAUHINIA platform by entering their name, email address, delivery address, password, and two active phone numbers. The registration process should be simple and straightforward. Customers should be able to check in to their accounts using their email address and password after registering. The system should ensure user data security and password security.
- **Browse and Search for Products:** Customers should be able to quickly navigate BAUHINIA's product catalog. The system should provide effective search and filtering features to assist customers in quickly and efficiently finding specific goods or product categories.
- **Check Product Availability:** Customers should be able to see a product's availability status in real-time when viewing it. If a product is out of stock, the system should notify the customer and possibly offer the option to be notified when it is back in stock.
- **Add Items to Cart:** Customers should be able to add items to an online shopping cart, selecting the quantity they want for each item. Before going through with the checkout, they ought to be able to review their shopping cart, change the quantities, or delete items.
- **Checkout Process:** Customers should get a summary of their order, including the total amount to be paid, during the checkout process. They should be able to confirm their billing information as well as select their chosen payment method. As BAUHINIA currently uses a cash-on-delivery system, this option should be clearly presented. The system should then confirm the successful placement of the order, providing the customer with an order reference number for tracking purposes.

- View and change Personal Information: Customers should be able to view and change their personal information in their account settings, including their delivery address and contact numbers. To protect client data, the system should ensure that any such updates are done securely.

The customer-facing aspects of the new system should focus on enhancing the user experience, promoting user engagement, and facilitating secure and efficient online shopping.

1.5.4 User Requirements for Inventory Handling Clerk

The inventory handling clerk plays a key role in BAUHINIA's operations, and the new system should provide a robust set of tools to facilitate the efficient management of inventory.

- User Registration and Sign-In: The registration and sign-in process for inventory handling clerks should be secure, just like it is for consumers. They should use their professional email address and a secure password to sign up. The system should offer different functionalities in accordance with the differences between customer and staff accounts.
- Add New Items to Inventory: The inventory handling clerk should have the ability to add new items to the inventory. This process should include adding detailed product information, such as item code, product description, size variations, price, and initial quantity. The system should validate these inputs to ensure accuracy and consistency in the inventory data.
- Update Existing Item Details: The clerk should be able to change existing inventory item data. This could include modifying the item's description, pricing, or available

quantities, as well as terminating it. Any updates should be reflected in real time in the customer-accessible product catalog.

- Access to Real-Time Inventory Data: The system should provide the clerk with real-time data on inventory levels. This should help the clerk maintain optimal stock levels, restocking items when necessary, and preventing overstock or understock situations. The system could provide alerts when inventory levels for certain items fall below a predefined threshold.
- View and Update Personal Information: Just like customers, inventory handling clerks should be able to view and update their personal information securely within the system.

By meeting these requirements, the new system will enable the inventory handling clerk to manage BAUHINIA's inventory more effectively and efficiently, contributing to smoother operations and better customer service.

1.5.5 User Requirements for Production Manager

The Production Manager at BAUHINIA plays a crucial role in ensuring the smooth flow of operations, from inventory management to order fulfillment. The new system should be equipped with features that can significantly streamline these processes and facilitate decision-making.

- User Registration and Sign-In: The Production Manager should be able to register and sign in to the system in a secure manner. Individual credentials should be unique, ensuring safe access to system functions relevant to their role.
- Generate Daily Orders Report: A crucial part of the Production Manager's role is the overview and management of the orders placed by the customers. The system should allow the Production Manager to generate daily reports outlining the orders placed, their status, and other related information. This data can help track sales

performance, identify any issues, and ensure that the order fulfillment process is running smoothly.

- Generate Daily Product Availability Report: Another critical function is the ability to generate product availability reports. This feature would make it possible for the Production Manager to assess current stock levels, spot products that might need to be ordered again soon, and make sure that high-demand items are always available. To ensure that these reports are accurate, the system should deliver real-time data.
- Access to Real-Time Inventory Data: In addition to generating reports, the system should provide the Production Manager with real-time access to inventory data. This allows for an immediate overview of the available stock and assists in making quick decisions regarding inventory management.
- View and Update Personal Information: Lastly, the Production Manager should be able to securely view and update their personal information within the system.

By fulfilling these requirements, the new system would provide valuable support to the Production Manager, promoting effective oversight and management of BAUHINIA's operations.

1.5.6 User Requirements for Chief Accountant

As the person responsible for the financial overview of BAUHINIA, the Chief Accountant needs specific tools and functionalities to effectively track, analyze, and report the company's financial performance.

- User Registration and Sign-In: The Chief Accountant should have secure access to the system, ensuring the security of sensitive financial data. They, like all other users, should register and sign in with unique credentials.
- Generate Monthly Income Report: An essential part of the Chief Accountant's role is the ability to generate detailed monthly income reports. The system should automatically gather and show important information, such as total sales, returns, discounts, and cash on delivery collected, in a format that enables for simple

analysis and reporting. Because these reports have a direct impact on the company's financial goals and decisions, the system should assure data integrity and accuracy.

- View and Update Personal Information: The Chief Accountant, like other users, should be able to securely view and update their personal information within the system.

By offering these functionalities, the new system may greatly simplify the Chief Accountant's workflow and guarantee that they have access to the data they need to maintain BAUHINIA's financial health. This also enhances overall business transparency because timely and accurate financial reporting are essential for strategic planning and top-level decision-making inside the organization.

1.5.7 Other User Requirements

In addition to the specific requirements of the various user roles, there are several general requirements that all users, regardless of their role, will need from the new system. These primarily revolve around usability, security, performance, and scalability.

- User-Friendly Interface: Whether a customer is looking for products or an employee is managing inventory, all users should find the system simple to access and use. The user interface should be simple to use minimizing the learning curve for new users and facilitating efficient use. Appearance should be considered as well, to ensure that the system is visually appealing and matches with BAUHINIA's brand image.
- Secure Access and Data Protection: Since the system handles sensitive user data, such as personal information and order details, comprehensive security measures are required. These could include data encryption, safe communication protocols, and frequent security assessments. User access should be controlled by unique credentials, and the system should log user activity for audit purposes.
- Reliable Performance and Availability: Users should be able to access the system whenever they need to, and it should respond quickly to their interactions. This

requires reliable hosting, efficient code, and regular maintenance to identify and resolve any performance issues. Downtime should be kept to a minimum to avoid disrupting BAUHINIA's activities and ensuring a positive customer experience.

- Scalability: As BAUHINIA grows, the system should be able to scale to handle more users and larger volumes of data. This requires a scalable architecture and planning for potential upgrades to system resources.
- Support and maintenance: Ongoing support and maintenance are critical to ensuring the system's continued operation and usefulness. This includes debugging and resolving difficulties, installing and upgrading software, and giving user assistance.

By meeting these general requirements, the new system can ensure a positive experience for all users while also maintaining the security, reliability, and scalability necessary for BAUHINIA's ongoing success.

1.5.8 System Requirements

The requirements of the proposed order tracking system for BAUHINIA can be broadly classified into two types: Functional Requirements and Non-Functional Requirements.

Functional Requirements:

Here is a summary chart of the functional requirements:

Table 1. 11 Functional Requirements

No.	Functional Requirement	Description
1	Customer Registration	Allows customers to create an account with necessary information.
2	Customer Sign-in	Allows customers to access their account using credentials.
3	Browse Products	Enables customers to view all available products.
4	Add to Cart	Customers can select products and add them to a shopping cart.
5	Checkout	Customers can review their cart, see the total amount, confirm billing details, and place the order.
6	Staff Registration	Permits authorized personnel to create staff accounts.
7	Staff Sign-in	Allows staff members to access the system with their credentials.
8	Daily Orders Report	Enables the Production Manager to generate a report of all orders placed on a particular day.
9	Daily Product Availability Report	Enables the Production Manager to generate a report of product availability.
10	Add/Update Inventory	Allows the Inventory Clerk to add new items and update existing items in the inventory.
11	Monthly Income Report	Enables the Chief Accountant to generate a report of the monthly income.

Non-Functional Requirements:

Here is a summary chart of the non-functional requirements:

Table 1. 12 Non-Functional Requirement

No.	Non-Functional Requirement	Description
1	Usability	The system should be user-friendly and intuitive.
2	Performance	The system should be able to handle high traffic and load.
3	Scalability	The system should be able to scale and accommodate growth.
4	Reliability	The system should have a high availability and low failure rate.
5	Security	The system should ensure data privacy and security.
6	Maintainability	The system should be easy to maintain and update.
7	Compatibility	The system should be compatible with multiple platforms and browsers.
8	Data Integrity	The system should maintain the accuracy and consistency of data.
9	Access Control	The system should ensure proper role-based access control.
10	Response Time	The system should provide a quick response to user actions.

These requirements will guide the software development process, ensuring that the end product effectively meets the needs of BAUHINIA's staff and customers.

1.5.9 Existing System Analysis

The existing system at BAUHINIA is a manual one with social media based offline transactions and interactions. Customers currently visit the physical store to browse/ order through social media, select, and purchase products. On the operational side, the staff maintains manual records for inventory, sales, and customer details. And also currently, they are handling orders through social media networks such as Facebook and Instagram. Customers can message BAUHINIA requesting an item/s by sending the item code, size and required quantity. If the item is available, the customer is required to send the delivery address, contact number to confirm the order.

The limitations of the current system are as follows:

- Limited Accessibility: The current system restricts the customers to visit the physical store for purchases/ order through social media, which limits accessibility for customers who are not in the vicinity or prefer online shopping due to convenience or other reasons.
- Inefficient Inventory Management: The current manual system of inventory management might result in inefficiencies such as errors in recording, stock mismanagement, and difficulty in tracking real-time inventory levels.
- Limited Customer Reach: As BAUHINIA only operates social media based offline /through social media, it restricts its customer base to the local area. It also misses out on potential customers who prefer online shopping.
- Manual Record Keeping: The staff maintains manual records for sales, inventory, and customer data. This can be time-consuming and prone to human errors. Also, analyzing this data for insights is tedious and challenging.
- Lack of Data Analysis: With the current system, analyzing sales trends, customer behavior, and inventory management is difficult due to the absence of digitized and structured data.
- Absence of Customer Profile: Since there is no dedicated online platform, BAUHINIA misses out on creating a customer profile that can aid in personalizing the shopping experience and improving customer loyalty.

Given these limitations, BAUHINIA aims to transition from a manual, social media based offline system to an automated, online platform. The objective is to provide customers with the convenience of shopping from anywhere at any time, enhance the shopping experience, manage inventory efficiently, reach a broader customer base, and use data for strategic business decisions.

The new system will include features such as product browsing, product details, shopping cart, checkout process, order tracking for customers, and login, sign-up, reports, inventory update, and dashboard for staff. These features are aimed at overcoming the limitations of the current system and enhancing BAUHINIA's business operations.

1.5.10 Proposed System Analysis

The proposed system for BAUHINIA is a comprehensive e-commerce solution that is designed to transition BAUHINIA's current manual, social media based offline operations to an automated, online platform. The system aims to increase operational efficiency, expand customer reach, and enhance the shopping experience.

The key components of the proposed system include:

- Customer Interface: The customers will have access to features such as account creation and login, product browsing, product details, shopping cart, checkout, and order tracking. This will allow customers to shop conveniently from anywhere, anytime, and keep track of their orders.
- Staff Interface: The staff will have access to features like account creation and login, generating reports, updating inventory, and a dashboard that provides an overview of business operations. This will enhance the staff's ability to manage inventory, keep track of sales, and make data-driven decisions.
- Product Management: The system will have an extensive database of products that can be easily managed and updated by the staff. Each product will have detailed information, including images, descriptions, prices, and availability status.

- Shopping Cart and Checkout: Customers will be able to add products to a virtual shopping cart and proceed to a seamless and secure checkout process, including various payment options.
- Order Management and Tracking: Post purchase, customers will be able to track the status of their orders, and staff can manage these orders efficiently.
- Report Generation: The system will include functionality to generate various reports like sales, inventory, and customer behavior, helping BAUHINIA make informed business decisions.
- Security and Compliance: The system will adhere to security standards and data protection laws to ensure safe transactions and protect customer data.

The proposed system is designed with an aim to overcome the limitations of the existing system, and provide a scalable solution that will support BAUHINIA's business growth and adapt to evolving business needs. By transitioning to this system, BAUHINIA can provide an enhanced shopping experience to its customers, manage operations more efficiently, and leverage data for strategic decision making.

1.5.11 Feasibility Analysis

Technical Feasibility: The proposed system for BAUHINIA is technically feasible. The development will utilize Python and Django, which are well-established and widely supported technologies suitable for web development. The system also relies on PostgreSQL as the database management system, which is efficient and reliable for handling large amounts of data. The technical skills needed for developing and maintaining the system are readily available in the market.

- 1) **Operational Feasibility:** The proposed system is designed to fit seamlessly into BAUHINIA's existing business operations, enhancing its efficiency and effectiveness. The system has an intuitive interface to ensure ease of use by both

staff and customers, reducing the learning curve. Moreover, necessary training will be provided to the staff to ensure a smooth transition to the new system.

- 2) Economic Feasibility: The cost of developing and maintaining the system is justifiable considering the significant benefits it will bring to BAUHINIA. These include increased sales due to wider customer reach, reduced operational costs due to automation, and enhanced decision-making due to better data analysis. Therefore, the system is expected to deliver a positive return on investment.
- 3) Legal Feasibility: The system will be designed to comply with all applicable laws and regulations, including data protection and privacy laws. Regular audits and updates will be carried out to ensure ongoing compliance.
- 4) Schedule Feasibility: Based on the project plan, the development and deployment of the system are expected to be completed within a reasonable timeline. This will be ensured through efficient project management and by mitigating potential risks that could cause delays.

In conclusion, the proposed system for BAUHINIA is feasible in terms of technical, operational, economic, legal, and schedule aspects. Implementing this system would help BAUHINIA achieve its goal of transforming its business operations and enhancing the shopping experience for its customers.

1.5.12 Operational Feasibility Analysis

Operational feasibility assesses how well the proposed system solves the problems at hand and how well it fits within the existing business environment.

- System Integration: The new system is designed to seamlessly integrate with BAUHINIA's existing operational procedures. This includes the user-friendly interfaces for both customers and staff, facilitating easier navigation and enhancing the overall user experience. The system can also be integrated with existing software, if any, minimizing disruptions to current operations.
- Ease of Use: The new system is designed with simplicity and user-friendliness in mind. Its interfaces, whether for customers browsing products or staff managing inventory, are intuitive and easy to navigate. Clear instructions and prompts will be provided where necessary. Training will also be provided to the staff to ensure a smooth transition.
- Workflow Improvement: The new system is expected to improve BAUHINIA's current workflow. Automated processes like inventory updates, order tracking, report generation, etc., will enhance efficiency, reduce manual errors, and free up staff time for more important tasks.
- User Acceptance: The proposed system has features that meet the needs of its potential users, i.e., the staff and the customers of BAUHINIA. Customers will benefit from the ease of browsing products, placing orders, and tracking their orders, thereby improving their shopping experience. The staff, on the other hand, will benefit from the streamlined processes and the availability of real-time data for decision making. With proper training and support, user acceptance is expected to be high.
- System Support and Maintenance: The proposed system is designed to be easily maintained and upgraded as necessary. Technical support will be available to troubleshoot any issues that arise, and system updates will be carried out regularly to ensure the system remains secure and up-to-date.

In conclusion, the proposed system is operationally feasible as it effectively addresses BAUHINIA's needs, fits well within its existing operational environment, and is expected to improve overall business efficiency and customer satisfaction.

1.5.13 Economic Feasibility Analysis

Economic feasibility involves a cost-benefit analysis of the proposed system, assessing whether the financial benefits outweigh the costs of development and maintenance.

- **Development Costs:** The development costs include expenses related to the design, coding, testing, and deployment of the system. This also includes the cost of the software development team, tools and technologies used, and the cost of any additional hardware required.
- **Maintenance Costs:** Post-deployment, there will be ongoing costs for maintaining and updating the system, providing user support, and managing system security. These costs should be included in BAUHINIA's annual IT budget.
- **Training Costs:** Training will be necessary for staff members to learn how to use and manage the new system. These costs include the development of training materials and potentially the hiring of a trainer or allocating time for existing IT staff to conduct the training.
- **Operational Costs:** These are the costs associated with the daily operation of the system, such as hosting costs, electricity, and any additional costs like third-party services.

Against these costs, we assess the following benefits:

- **Increased Efficiency:** The proposed system automates several processes, such as inventory updates and order tracking, reducing manual work, and saving staff time. This could lead to significant cost savings in the long run.

- Improved Customer Satisfaction: The new system will enhance the customer shopping experience by making product browsing, purchasing, and tracking easier and more efficient. This could potentially increase sales and customer retention.
- Improved Decision Making: The system's report generation feature will provide useful insights into sales, inventory, and other business aspects, aiding in better decision making.
- Reduced Errors: Automation of processes will minimize human errors, reducing the costs associated with such errors, including customer dissatisfaction and the need for corrections or refunds.

In conclusion, while the proposed system involves significant upfront and ongoing costs, the benefits in terms of increased efficiency, customer satisfaction, and improved decision-making make it economically feasible in the long run. A more detailed cost-benefit analysis should be conducted to quantify these costs and benefits and determine the expected return on investment.

1.5.14 Legal Feasibility Analysis

Legal feasibility examines the legal issues that the proposed system may encounter. In the case of the BAUHINIA system, it is critical to comply with all applicable laws and regulations related to e-commerce, data protection, and online business operations.

- Data Protection Laws: BAUHINIA's system collects, stores, and processes customer data. Therefore, it must comply with data protection laws and regulations, such as the General Data Protection Regulation (GDPR) if it operates within the EU, or equivalent laws in other jurisdictions. It is essential to ensure that the system includes adequate measures for data privacy and security, including obtaining informed consent from users for data collection and providing mechanisms for data access, rectification, and erasure.
- E-Commerce Regulations: Being an e-commerce platform, BAUHINIA needs to comply with relevant e-commerce laws, including those related to online sales, digital marketing, and consumer rights. This includes providing clear and accurate product information, secure payment methods, and a straightforward process for returns and refunds.
- Cybersecurity Laws: With the rise in cybersecurity threats, various jurisdictions have laws related to cybersecurity that businesses must comply with. BAUHINIA needs to ensure that its system includes robust security measures to protect against cyber threats and comply with any reporting or other obligations in the event of a data breach.
- Accessibility Laws: To ensure that the system is accessible to all users, including those with disabilities, it should comply with relevant accessibility standards such as the Web Content Accessibility Guidelines (WCAG).

Based on the proposed system design and the discussions so far, it appears that BAUHINIA is planning to incorporate measures to address these legal requirements. However, a detailed legal review should be undertaken, and regular audits should be conducted to ensure ongoing compliance. It would be advisable to consult with a legal expert

specializing in e-commerce and data protection to ensure that all legal aspects are adequately covered.

1.5.15 Schedule Feasibility Analysis

Schedule feasibility assesses whether the proposed system can be developed and implemented within the given timeline. For the BAUHINIA system, this would involve considering the complexity of the system, the number of functionalities to be implemented, the resources available (including development team members and their skills), and any constraints or dependencies that might impact the schedule.

Based on the complexity and scope of the BAUHINIA system, as well as the methodologies and strategies discussed so far, the following key stages and approximate timelines could be considered:

- System Analysis and Design: This includes understanding the system requirements, creating the system design, and planning the development process. This stage could take around 2-3 months.
- Coding and Implementation: This stage involves coding the system functionalities, setting up the database, and integrating different components. Given the complexity of the system and the number of functionalities to be implemented, this stage could take around 6-8 months.
- Testing: This includes unit testing, integration testing, system testing, and user acceptance testing. Depending on the issues encountered and the time required to fix them, this stage could take around 2-3 months.
- Deployment and Maintenance: This stage involves deploying the system in the production environment, training users, and providing ongoing maintenance and support. This stage could take around 1-2 months initially for deployment and training, followed by ongoing maintenance.

In total, considering the above stages, the development and implementation of the BAUHINIA system could take approximately 11-16 months. However, these are rough estimates and the actual timeline could vary depending on various factors.

Therefore, it's essential to have a detailed project plan with milestones and deadlines, to track the progress and manage any risks or delays effectively. Regular reviews and updates to the schedule may also be necessary as the project progresses.

Based on the discussions so far, it appears that BAUHINIA has a reasonable understanding of the development process and the resources required, which suggests that the proposed system is likely to be schedule feasible. However, it's crucial to manage the project effectively and be prepared to adjust the plan as needed to address any challenges or changes in scope that may arise.

1.5.16 System Development Methodology in Use

In developing the BAUHINIA system, we have chosen to use the Agile development methodology, specifically Scrum, due to its iterative approach and its ability to adapt to changes. Scrum focuses on developing a system in small, manageable chunks called as "sprints," which allows for design flexibility and quick adaptability to change. This strategy is appropriate for BAUHINIA because it enables ongoing system evaluation and improvement throughout the development process.

Agile Over Waterfall:

The Agile approach was preferred over the Waterfall model for its flexibility and adaptability. The Waterfall model is a sequential, linear process in which development proceeds in a single direction—downward, like a waterfall—through the stages of conception, initiation, analysis, design, construction, testing, production, implementation, and maintenance.

While the Waterfall model works effectively for projects with clear, unchanging needs, it is less effective for BAUHINIA, where requirements may change as the project advances. Agile, on the other hand, embraces change and encourages iterative development and regular feedback.

The Agile model is intended to manage project difficulties by dividing the project down into manageable pieces of work, or "iterations." Each iteration will result in a working product increment, making it possible to deliver working software from an early stage of the project, and continuously improving it through subsequent iterations. This continual feedback and flexibility enables Agile to provide a product that better meets the demands of the client, even if those needs change during the development process.

Why Scrum:

Scrum is a type of Agile methodology known for its simplicity, flexibility, and effectiveness. The choice of Scrum was made due to the following reasons:

- Iterative Development: Scrum divides the project into manageable 'sprints' that typically span 2-4 weeks. This enables the team to focus on a small set of features during each sprint and to adjust plans for the next sprint depending on feedback and learnings from the previous one.
- Roles and Responsibilities: The Product Owner, Scrum Master, and Development Team all have distinct roles in Scrum. This provides a balance between business objectives (controlled by the Product Owner), team well-being and process optimization (managed by the Scrum Master), and the creation of high-quality increments (managed by the Development Team).
- Frequent Inspections and Adaptations: Scrum events (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) create regular opportunities to inspect and adapt the product, the team, and the working practices. This ensures that the project is constantly improved.
- Transparency and Collaboration: Scrum encourages a high level of transparency and collaboration inside the team as well as with stakeholders. This ensures that everyone has a clear knowledge of the product's progress and the team's issues.

Scrum is an excellent solution for BAUHINIA because of its iterative approach and emphasis on inspection, adaptability, and quick feedback loops. It will help in managing the project's complexities and uncertainties, while also ensuring that the team delivers a product that meets the demands and expectations of the stakeholders.

1.5.17 System Design

System design is an important stage in which we construct the blueprint for the proposed system. We define the system architecture, models, and strategies for data, interface, and security design at this stage. We used a variety of strategies and methodologies when designing the BAUHINIA system to achieve a thorough and efficient design.

1) Use Case Design

The BAUHINIA system use case design entails determining all of the interactions that users (Customers, Inventory handling Clerks, Production Managers, Chief Accountants) will have with the system. Each user role has particular use cases associated with their tasks. These use cases help to see system functionality from the perspective of the users, allowing for a more user-centric design.

2) ERD Design

Entity Relationship Diagram (ERD) is a valuable tool for developing the system's database schema. The ERD for BAUHINIA represents all data entities and their relationships (such as Customers, Orders, Products, Inventory, and so on). The ERD assists in the visualization of data flow within the system, allowing for more effective database architecture.

3) Class Diagram Design

Class diagrams are a part of the Unified Modeling Language (UML) and provide a static structure of the proposed system, including its classes, attributes, methods, and relationships among objects. They serve as a blueprint for the development of the software program.

4) Sequence Diagram Design

Sequence diagrams demonstrate the interaction between objects in the sequential order that those interactions occur. This visualization will be useful in understanding how system processes unfold during execution and how different elements interact with each other.

5) User Interface Design

The design of the user interface for the BAUHINIA system include producing a simple and user-friendly interface. The design prioritizes simplicity and usability, ensuring that all user groups may easily navigate and complete their responsibilities within the system. This includes, among other things, designing the system's landing page, product browsing page, checkout page, and admin dashboard.

6) Database Design

Database design involves creating a structured data model for the system's data requirements. For BAUHINIA, this would include designing tables for Users, Products, Orders, and Inventory, among others. Proper database design is crucial for ensuring data integrity, security, and performance of the system.

1.5.18 System Design – ERD Diagram

This Entity Relationship Diagram (ERD) captures the most critical entities and relationships in the BAUHINIA system, taking into account the different roles of the users and their actions within the system.

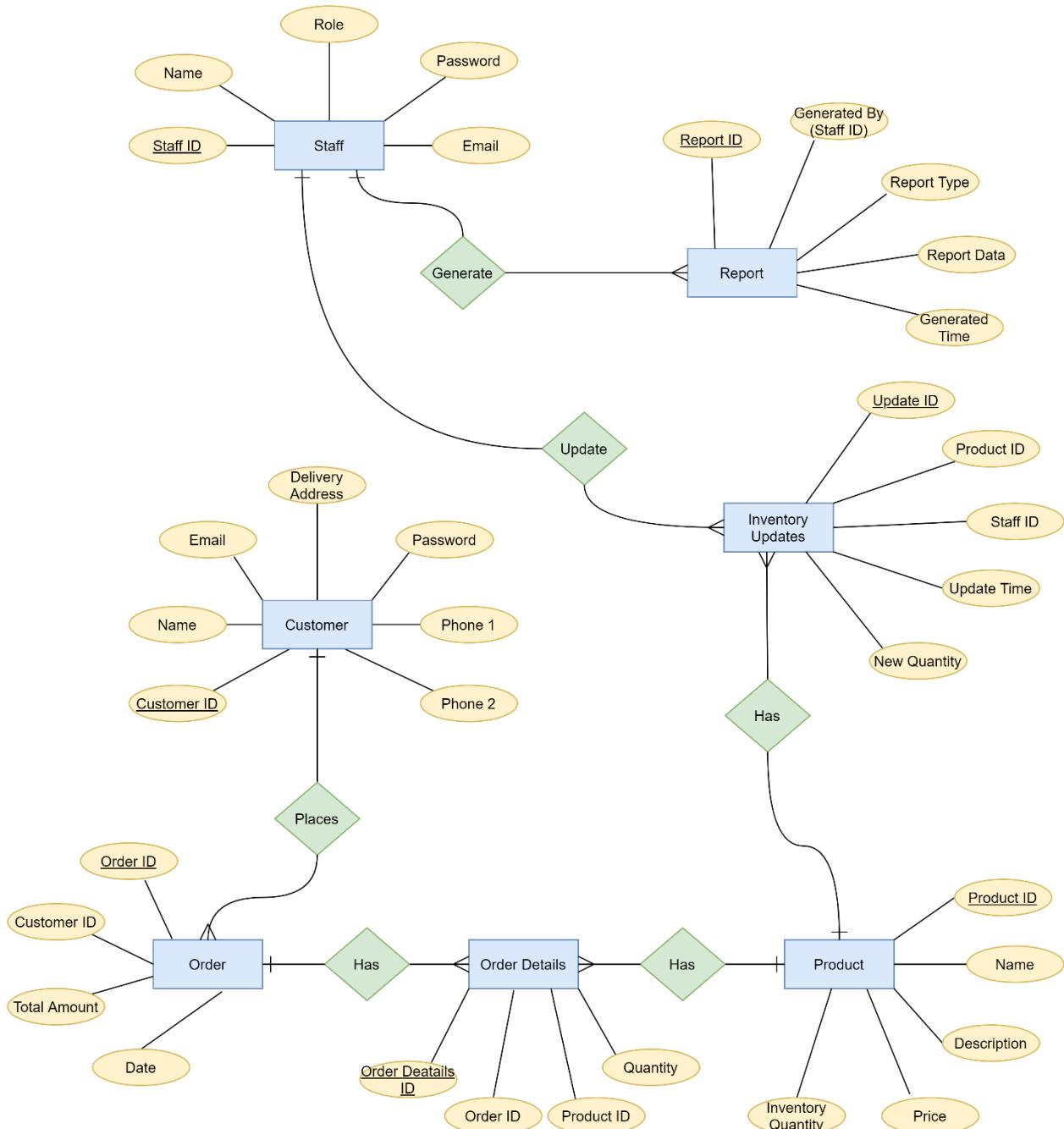


Figure 1. 1 ERD Diagram

Here are the relationships between each entity as described in the ERD:

- Customers to Orders: This is a one-to-many relationship. One customer can place multiple orders, but each order can only be associated with one customer. The CustomerID in the Orders entity is a foreign key that links to the CustomerID in the Customers entity.
- Orders to OrderDetails: This is also a one-to-many relationship. One order can contain multiple order details (i.e., multiple products), but each order detail is associated with only one order. The OrderID in the OrderDetails entity is a foreign key that links to the OrderID in the Orders entity.
- OrderDetails to Products: This is a many-to-one relationship. Multiple order details can refer to the same product, but each order detail refers to only one product. The ProductID in the OrderDetails entity is a foreign key that links to the ProductID in the Products entity.
- Products to Inventory Updates: This is a one-to-many relationship. One product can have multiple inventory updates, but each inventory update refers to only one product. The ProductID in the InventoryUpdates entity is a foreign key that links to the ProductID in the Products entity.
- Staff to Inventory Updates: This is a one-to-many relationship. One staff member can make multiple inventory updates, but each inventory update is made by only one staff member. The StaffID in the Inventory Updates entity is a foreign key that links to the StaffID in the Staff entity.
- Staff to Reports: This is a one-to-many relationship. One staff member can generate multiple reports, but each report is generated by only one staff member. The GeneratedBy in the Reports entity is a foreign key that links to the StaffID in the Staff entity.

1.5.19 System Design – Class Diagram

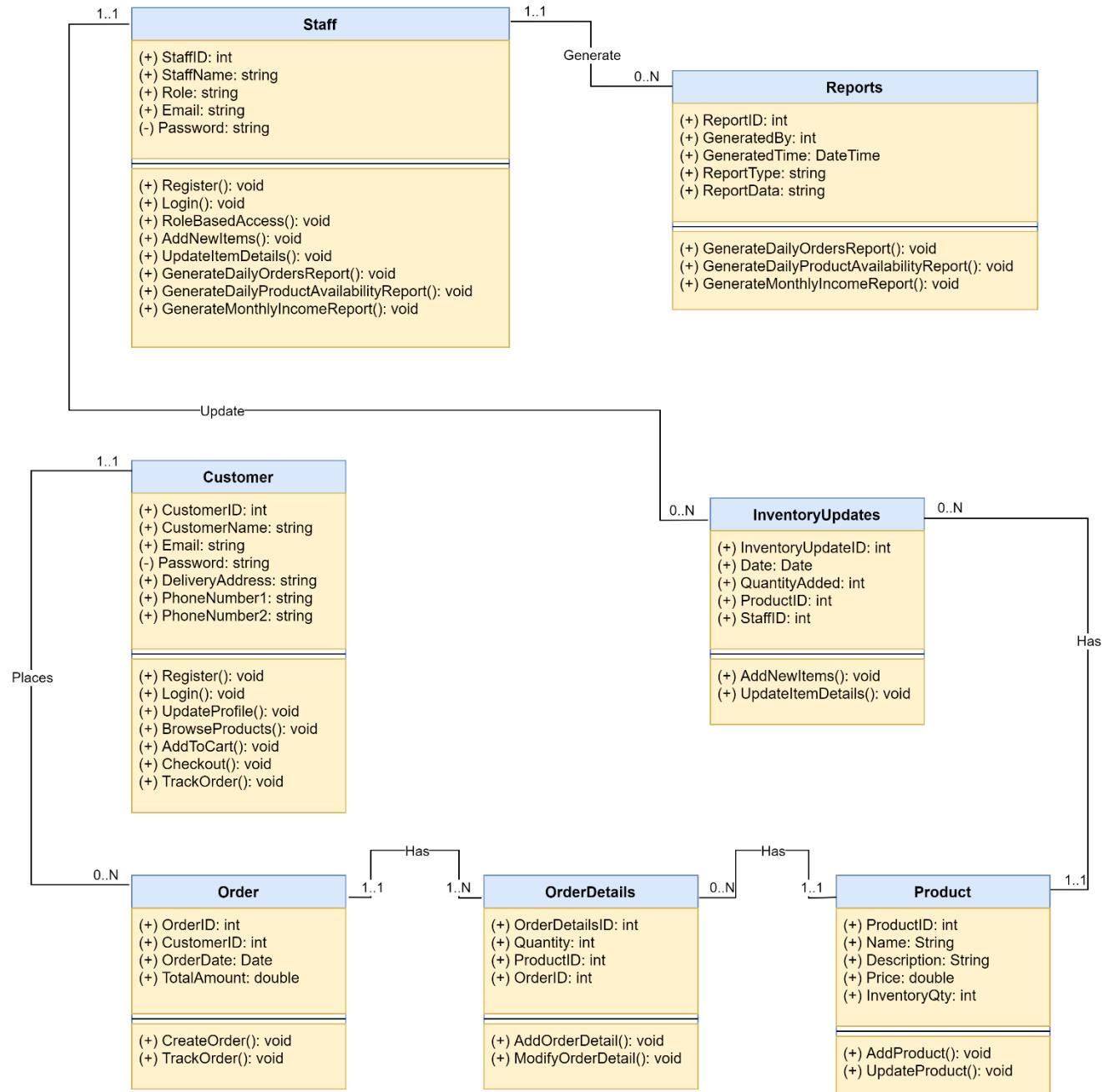


Figure 1. 2 Class Diagram

Here are the relationships between each class as described in the Class Diagram:

- Customer to Order: A customer can place zero, one, or many orders. Each order is associated with one and only one customer. This is a one-to-many relationship from the Customer to the Order class.
- Order to OrderDetails: An order can have one or many order details, but each order detail is associated with one and only one order. This is a one-to-many relationship from the Order to the OrderDetails class.
- OrderDetails to Product: Each order detail is associated with one and only one product. But one product can be part of zero, one, or many order details. This is a many-to-one relationship from the OrderDetails to the Product class.
- Product to InventoryUpdates: A product can have zero, one, or many inventory updates. Each inventory update is associated with one and only one product. This is a one-to-many relationship from the Product to the InventoryUpdates class.
- Staff to InventoryUpdates: A staff member can perform zero, one, or many inventory updates. Each inventory update is performed by one and only one staff member. This is a one-to-many relationship from the Staff to the InventoryUpdates class.
- Staff to Reports: A staff member can generate zero, one, or many reports. Each report is generated by one and only one staff member. This is a one-to-many relationship from the Staff to the Reports class.

In the class diagram I've provided, there isn't any explicit inheritance or superclass/subclass relationship. However, considering the nature of the system, I can recreate the Class Diagram with inheritance in certain areas.

One possible example of this could be:

User (Superclass) → Customer and Staff (Subclasses): Both Customer and Staff share common attributes like Name, Email, and Password, and behaviors such as Register and Login. They could both inherit these from a User superclass, and then each add their own specific attributes and methods. This would make sense if other common behaviors between customers and staff members were anticipated in future development.

But, inheritance should be used carefully and with purpose, it's not always the best solution and can sometimes introduce more complexity than it solves. Other object-oriented concepts like composition or interface implementation might be better fits depending on the system's specific requirements and design.

1.5.20 System Design – Use Case Diagram

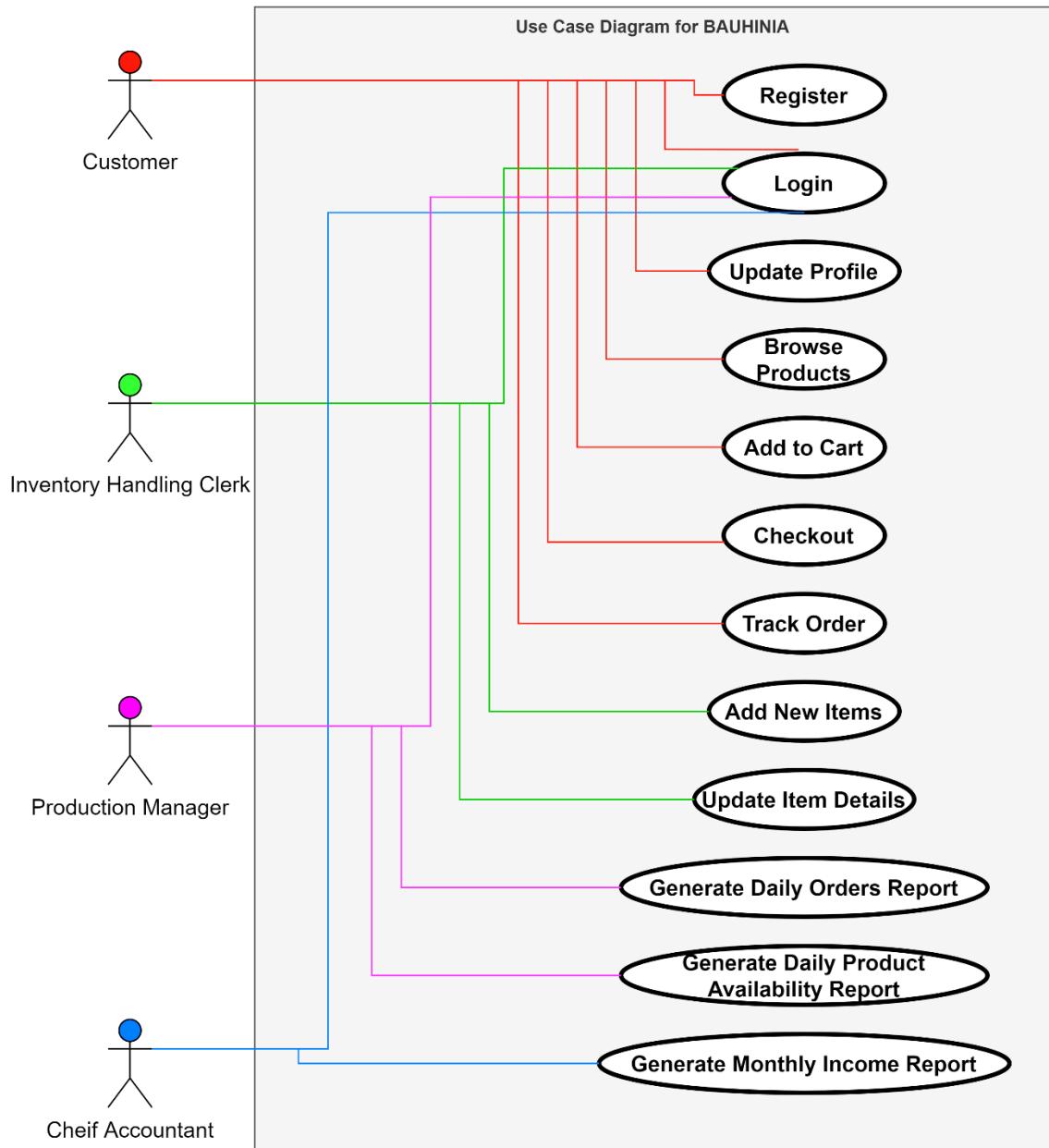


Figure 1. 3 Use Case Diagram

Here are the relationships between each Actor and Use Case as described in the Use Case:

- Customer - "Register", "Login", "Update Profile", "Browse Products", "Add to Cart", "Checkout", "Track Order"
- Inventory Handling Clerk - "Login", "Add New Items", "Update Item Details"
- Production Manager - "Login", "Generate Daily Orders Report", "Generate Daily Product Availability Report"
- Chief Accountant - "Login", "Generate Monthly Income Report"

In the case of "CreateOrder()" and "TrackOrder()" from the Order class, these operations correspond to the "Checkout" and "Track Order" use cases, respectively, for the Customer actor. When a Customer "Checks out", they are effectively creating an order, and when they "Track Order", they are using the "TrackOrder()" operation.

For the "AddOrderDetail()" and "ModifyOrderDetail()" operations from the OrderDetails class, these are typically included as part of the "Add to Cart" and "Checkout" use cases for the Customer actor. When a Customer adds a product to their cart, they are effectively adding an order detail, and any modifications to the products in their cart before checking out could be considered modifications of the order details.

Use Case Textual Descriptions:

- Register: Actor (Customer/Staff) inputs their details like name, email, password (and role for staff) to register into the system. System validates the information and creates a new account.
- Login: Actor (Customer/Staff) enters their email and password to login. System validates the credentials and if valid, grants access to the system.
- Update Profile: Actor (Customer) modifies their profile information such as delivery address or phone number. System updates the information in the database.

- Browse Products: Actor (Customer) views the list of all available products. System retrieves and displays product information from the database.
- Add to Cart: Actor (Customer) adds desired products to their cart. System updates the cart contents.
- Checkout: Actor (Customer) reviews the cart and places the order. System calculates the total amount, updates the order information in the database, and provides an order confirmation to the customer.
- Track Order: Actor (Customer) checks the status of their orders. System retrieves and displays the order status information from the database.
- Add New Items: Actor (Inventory Handling Clerk) adds new items to the inventory. System updates the inventory information in the database.
- Update Item Details: Actor (Inventory Handling Clerk) modifies the details of existing items in the inventory. System updates the inventory information in the database.
- Generate Daily Orders Report: Actor (Production Manager) requests a report of all orders for a given day. System retrieves the order information from the database and generates the report.
- Generate Daily Product Availability Report: Actor (Production Manager) requests a report of product availability. System retrieves the inventory information from the database and generates the report.
- Generate Monthly Income Report: Actor (Chief Accountant) requests a report of income for a specific month. System retrieves the relevant order and payment information from the database and generates the report.

1.5.21 System Design – Sequence Diagrams

Scenario 1: Customer Places an Order

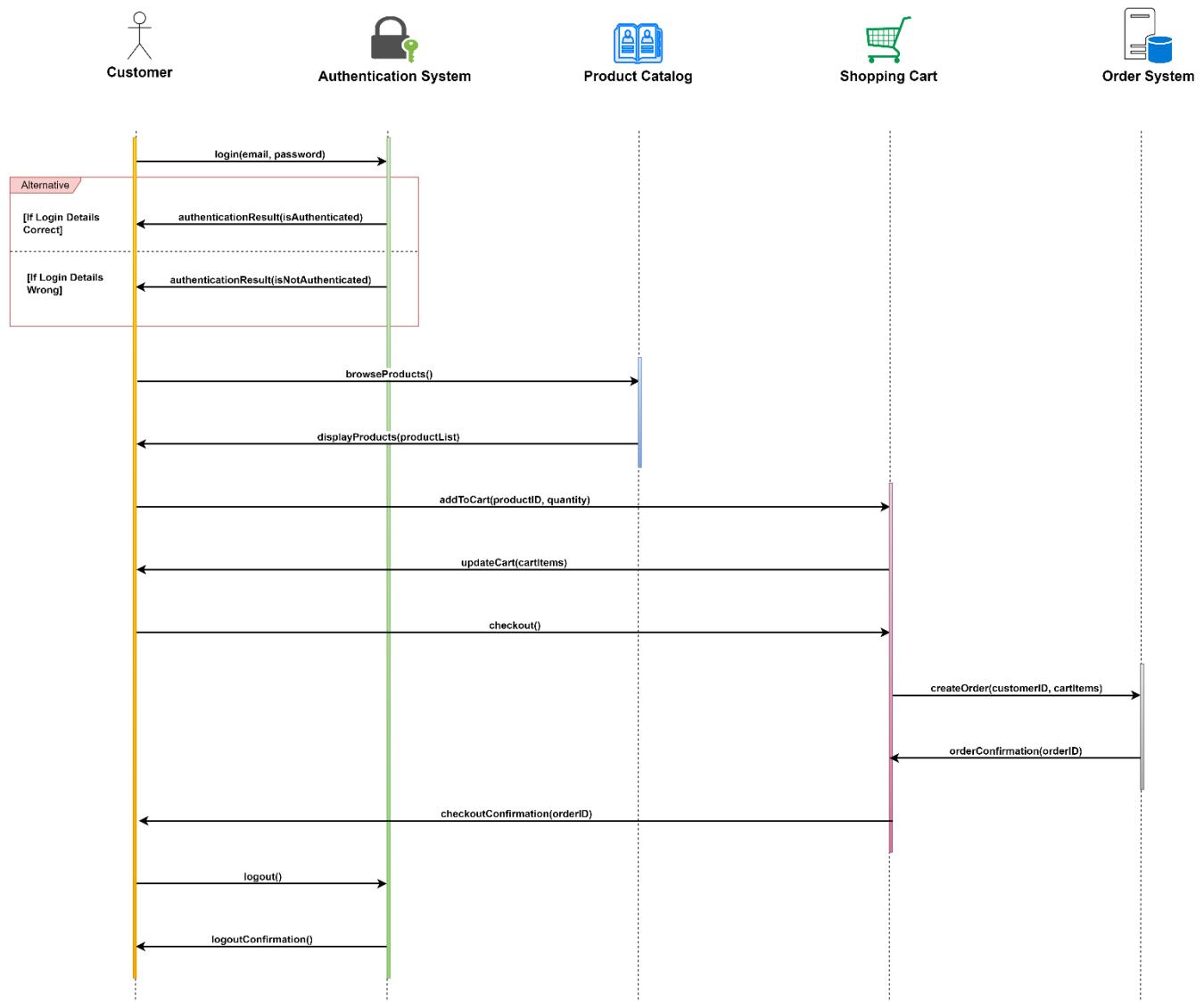


Figure 1. 4 Scenario 1: Customer Places an Order

Sequence of Interactions:

- a. The **Customer** object sends a message **login(email, password)** to the **Authentication System** object.
- b. **Authentication System** object checks the credentials and returns a message **authenticationResult(isAuthenticated)** to the **Customer**.
- c. If authenticated, the **Customer** object sends a message **browseProducts()** to the **Product Catalog** object.
- d. **Product Catalog** object fetches the product list and sends a message **displayProducts(productList)** back to the **Customer**.
- e. The **Customer** object selects a product and sends a message **addToCart(productId, quantity)** to the **Shopping Cart** object.
- f. The **Shopping Cart** object updates the cart and sends a message **updateCart(cartItems)** back to the **Customer**.
- g. The **Customer** object decides to checkout and sends a message **checkout()** to the **Shopping Cart** object.
- h. **Shopping Cart** object sends a message **createOrder(customerID, cartItems)** to the **Order System**.
- i. **Order System** object creates an order and sends a message **orderConfirmation(orderID)** back to the **Shopping Cart** object.
- j. The **Shopping Cart** object sends the confirmation message **checkoutConfirmation(orderID)** to the **Customer**.
- k. Finally, **Customer** object sends a message **logout()** to the **Authentication System** object and the **Authentication System** confirms the logout with a message **logoutConfirmation()** to the **Customer**.

Scenario 2: Staff Updates Inventory

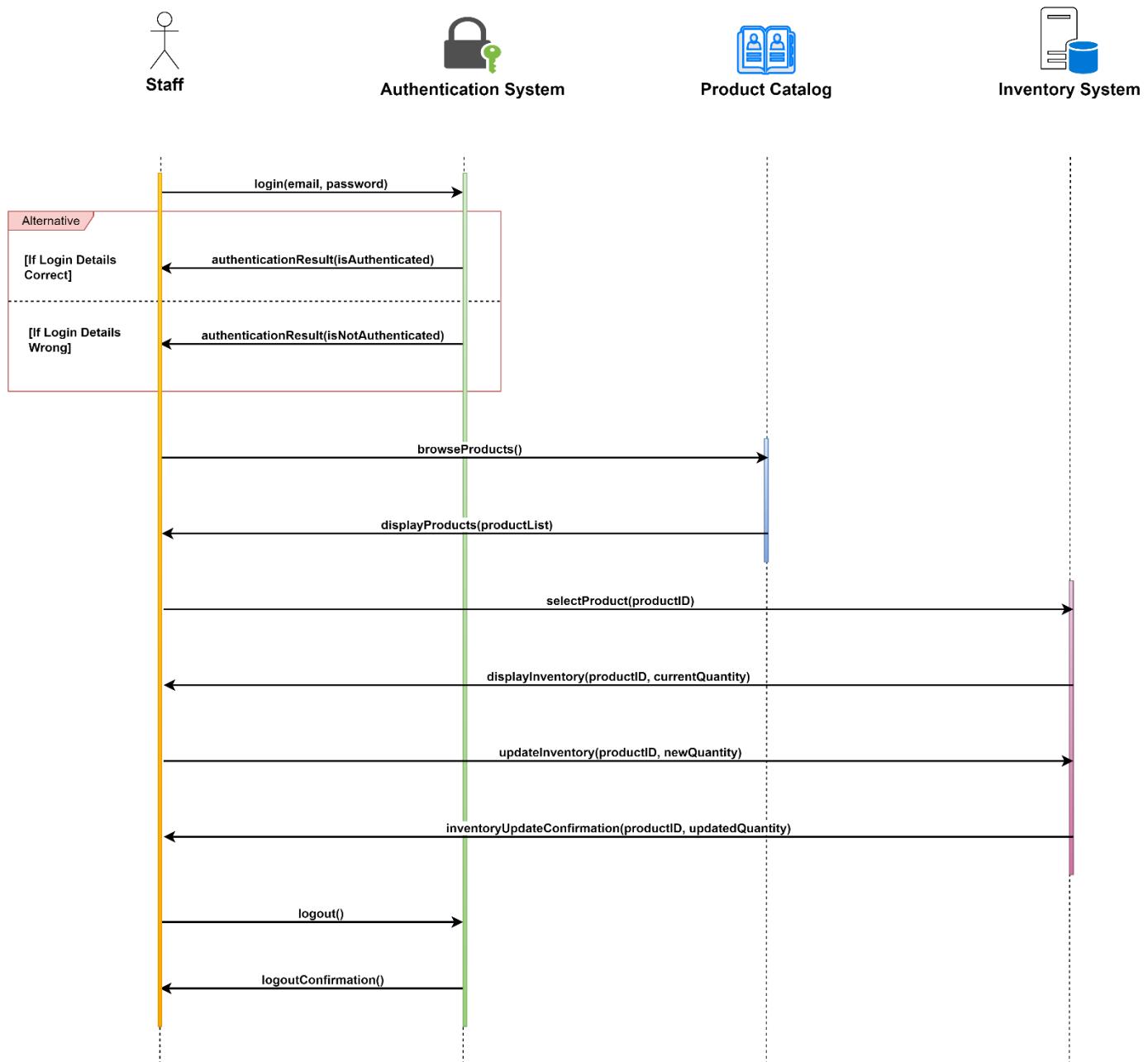


Figure 1. 5 Scenario 2: Staff Updates Inventory

Sequence of Interactions:

- a. The **Staff** object sends a message **login(email, password)** to the **Authentication System** object.
- b. **Authentication System** object checks the credentials and returns a message **authenticationResult(isAuthenticated)** to the **Staff**.
- c. If authenticated, the **Staff** object sends a message **browseProducts()** to the **Product Catalog** object.
- d. **Product Catalog** object fetches the product list and sends a message **displayProducts(productList)** back to the **Staff**.
- e. The **Staff** object selects a product and sends a message **selectProduct(productID)** to the **Inventory System** object.
- f. The **Inventory System** object fetches the current inventory details for the selected product and sends a message **displayInventory(productID, currentQuantity)** back to the **Staff**.
- g. The **Staff** object updates the inventory by sending a message **updateInventory(productID, newQuantity)** to the **Inventory System** object.
- h. **Inventory System** object updates the inventory and sends a confirmation message **inventoryUpdateConfirmation(productID, updatedQuantity)** back to the **Staff**.
- i. Finally, **Staff** object sends a message **logout()** to the **Authentication System** object and the **Authentication System** confirms the logout with a message **logoutConfirmation()** to the **Staff**.

Scenario 3: Customer Tracks an Order

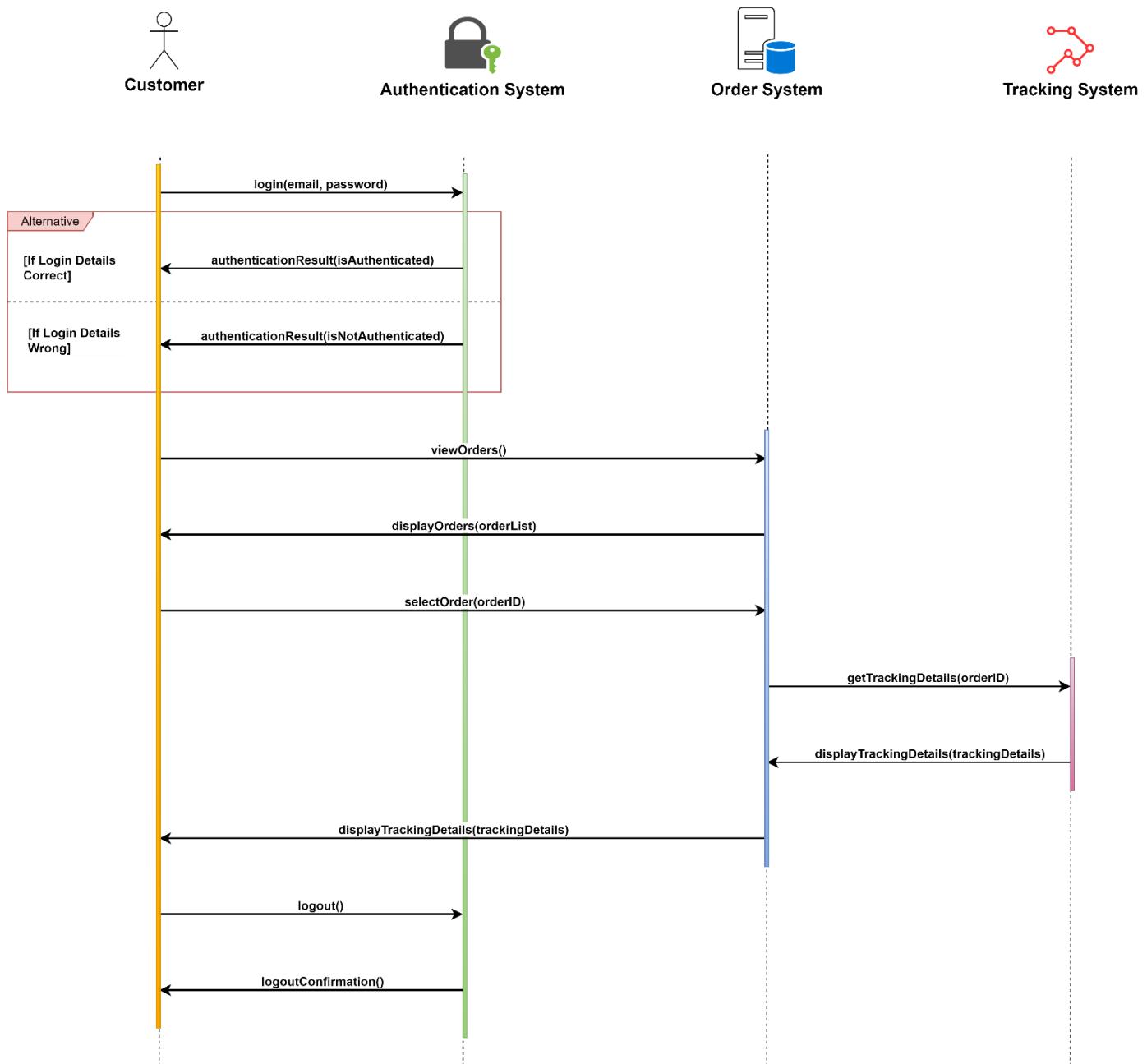


Figure 1. 6 Scenario 3: Customer Tracks an Order

Sequence of Interactions:

- a. The **Customer** object initiates the process by sending a **login(email, password)** message to the **Authentication System** object.
- b. The **Authentication System** object verifies the credentials and returns an **authenticationResult(isAuthenticated)** message to the **Customer**.
- c. If the customer is authenticated, the **Customer** sends a **viewOrders()** message to the **Order System** object.
- d. The **Order System** fetches the list of orders associated with the customer and sends a **displayOrders(orderList)** message back to the **Customer**.
- e. The **Customer** selects an order for tracking and sends a **selectOrder(orderID)** message to the **Order System**.
- f. The **Order System** forwards the selected orderID to the **Tracking System** object by sending a **getTrackingDetails(orderID)** message.
- g. The **Tracking System** object fetches the tracking details for the selected order and sends a **displayTrackingDetails(trackingDetails)** message back to the **Order System**.
- h. The **Order System** then forwards the trackingDetails to the **Customer** by sending a **displayTrackingDetails(trackingDetails)** message.
- i. Finally, the **Customer** sends a **logout()** message to the **Authentication System** object, and the **Authentication System** confirms the logout by sending a **logoutConfirmation()** message to the **Customer**.

Scenario 4: Staff Generates Report

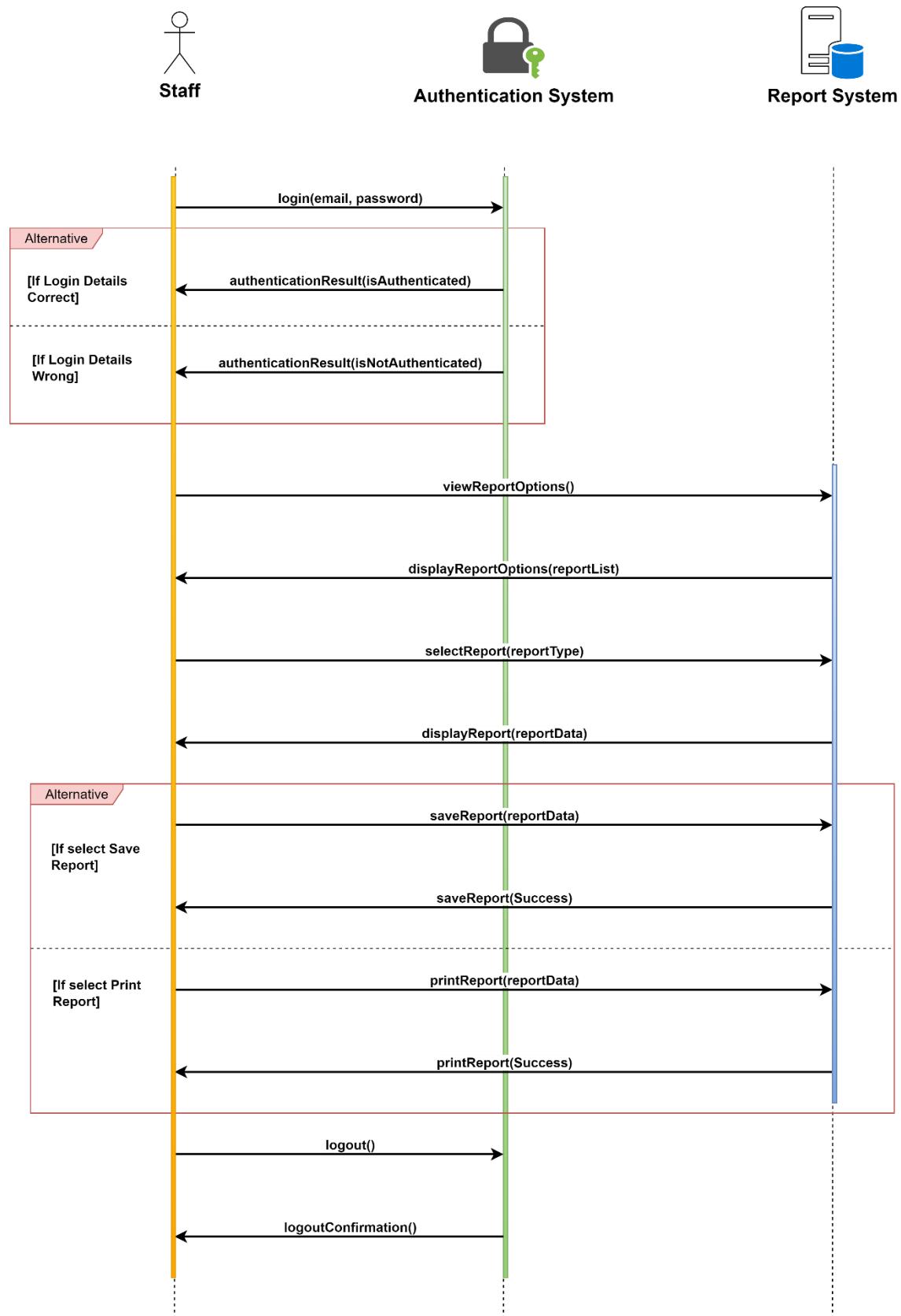


Figure 1. 7 Scenario 4: Staff Generates Report

Sequence of Interactions:

- a. The **Staff** object initiates the process by sending a **login(email, password)** message to the **Authentication System** object.
- b. The **Authentication System** object verifies the credentials and returns an **authenticationResult(isAuthenticated)** message to the **Staff**.
- c. If the staff member is authenticated, the **Staff** sends a **viewReportOptions()** message to the **Report System** object.
- d. The **Report System** presents the list of available report types and sends a **displayReportOptions(reportList)** message back to the **Staff**.
- e. The **Staff** selects a report type and sends a **selectReport(reportType)** message to the **Report System**.
- f. The **Report System** generates the selected report and sends a **displayReport(reportData)** message back to the **Staff**.
- g. The **Staff** can then decide to save or print the report by sending a **saveReport(reportData)** or **printReport(reportData)** message to the **Report System**.
- h. Finally, the **Staff** sends a **logout()** message to the **Authentication System** object, and the **Authentication System** confirms the logout by sending a **logoutConfirmation()** message to the **Staff**.

1.5.22 System Design – User Interfaces Design

The following pages I've designed for BAUHINIA company's system:

Customer - Login Page:

A page where existing customers can enter their email and password to log into their accounts.

Customer - Sign-up Page:

A page where new customers can create an account by providing their personal information such as name, email, and delivery address and by setting up a password.

Customer - Home Page:

The landing page a customer sees after logging into their account. It might feature popular products, personalized recommendations, ongoing deals, etc.

Customer – Products Browse Page:

A page that displays all available products that the customer can browse through. It can have filter and sort features for the customers to find specific products.

Customer – Products Details Page:

A page that gives detailed information about a specific product when a customer clicks on it from the browse page. This page will display product images, descriptions, reviews, etc.

Customer – Checkout Page:

This is the page where customers can finalize their order. They can review their cart, add shipping details, choose a payment method, and place the order.

Customer – Shopping Cart Page:

A page that shows all the products that the customer has added to their cart. It allows for the adjustment of quantity, removal of items, and review of the total cost.

Customer – Order Tracking Page:

This page allows customers to track their orders. It shows the progress of the order from the warehouse to delivery.

Staff – Login Page:

A page where staff members can log into their accounts using their credentials.

Staff – Sign-up Page:

A page for new staff members to create their accounts by providing necessary details like their name, role, and email and setting up a password.

Staff – Reports Page:

A page that allows staff to generate and review various reports such as sales reports, inventory reports, etc.

Staff – Inventory Update Page:

A page that allows staff to update the inventory as new stock arrives or existing stock is sold. They can modify the quantities of the products in the inventory. And they can see a more details about inventory through charts and tables.

Staff – Dashboard Page:

The main page for staff members after login. It could display summary information, important notifications, tasks, etc. based on their role.

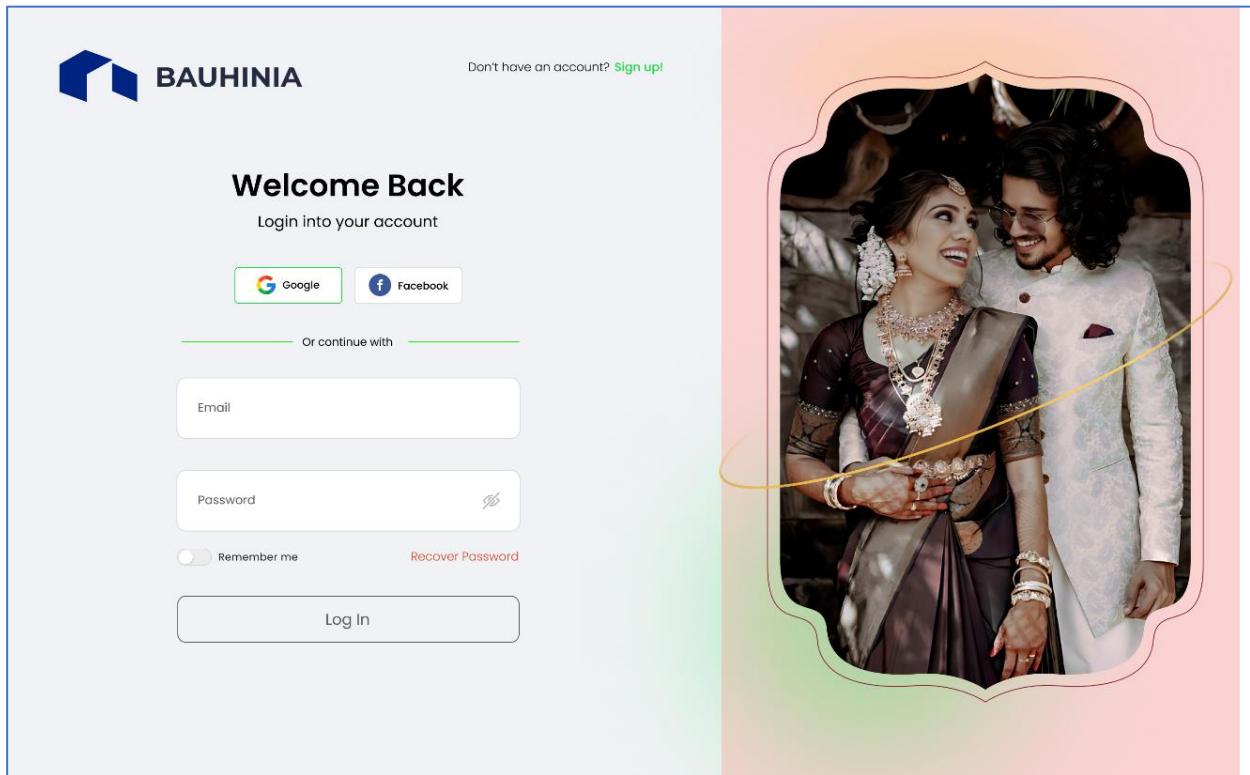
Customer - Login Page:

Figure 1. 8 Customer - Login Page

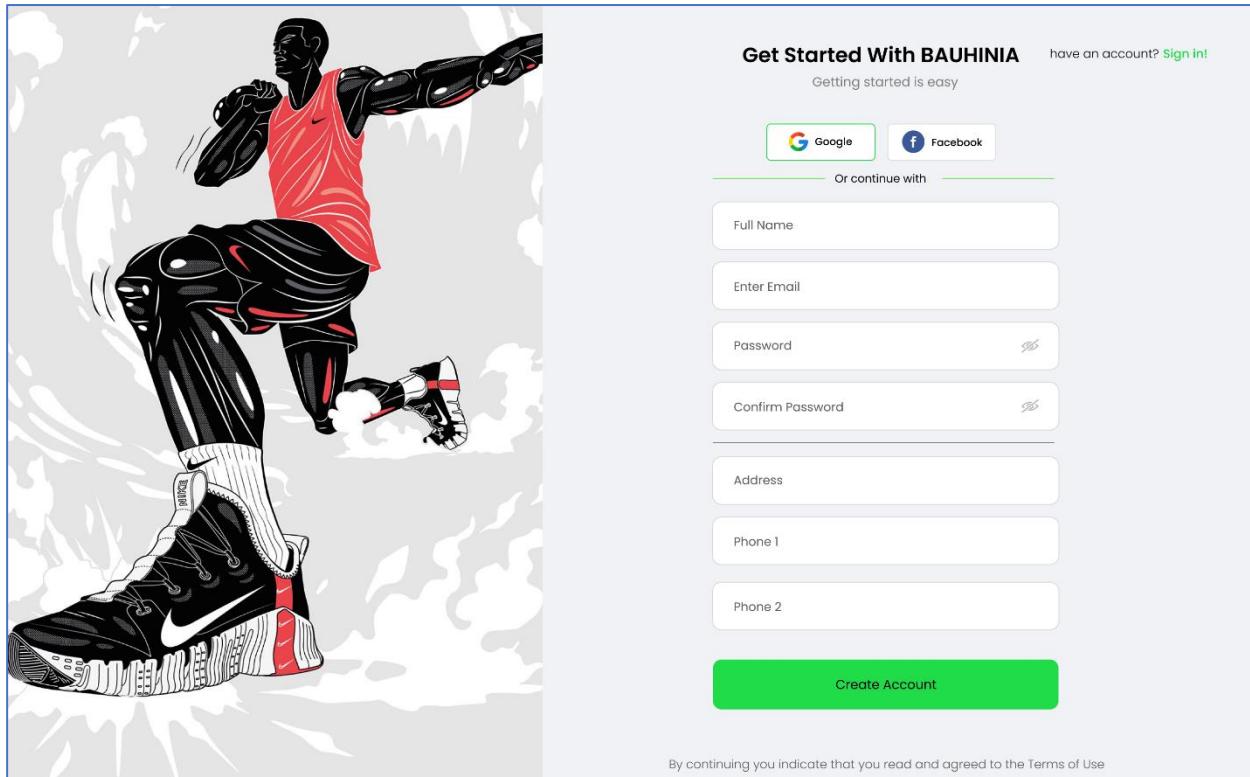
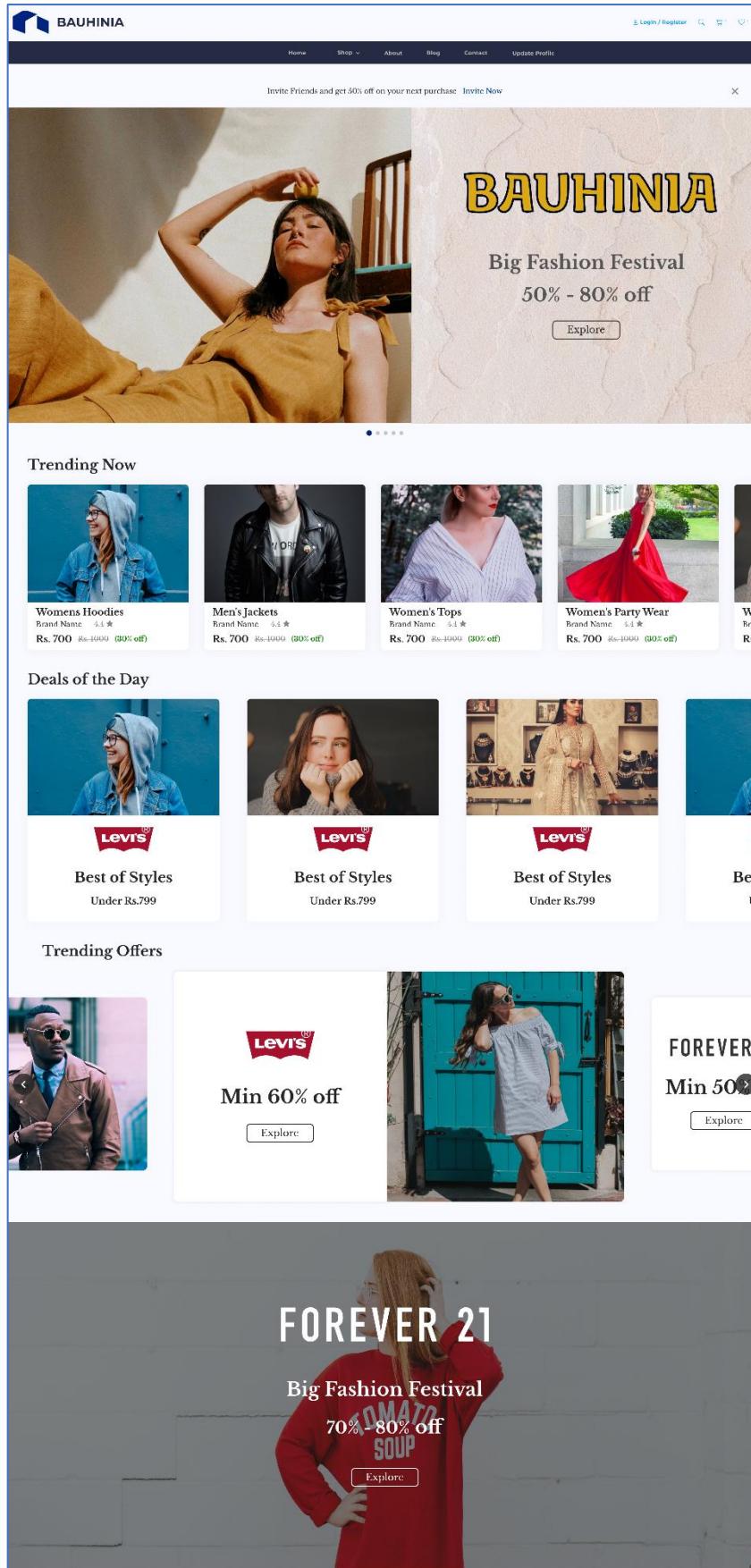
Customer - Sign-up Page:

Figure 1. 9 Customer - Sign-up Page

Customer - Home Page:



The screenshot shows the homepage of the BAUHINIA e-commerce platform. At the top, there is a navigation bar with links for Home, Shop, About, Blog, Contact, and Update Profile. A promotional banner on the right side of the header offers 50% off on invites. Below the header, a large image of a woman in a yellow dress is displayed, with the BAUHINIA logo and "Big Fashion Festival" text overlaid. A "Explore" button is present in the banner area. The main content area features several sections: "Trending Now" with four product cards (Womens Hoodies, Men's Jackets, Women's Tops, Women's Party Wear), "Deals of the Day" with three product cards (Best of Styles under Rs.799), and "Trending Offers" with three promotional boxes (Levi's Min 60% off, FOREVER 21 Min 50% off, and a third box partially visible). A large banner at the bottom for FOREVER 21 also features a woman in a red dress.

Shop by Categories



What Our Customer Says

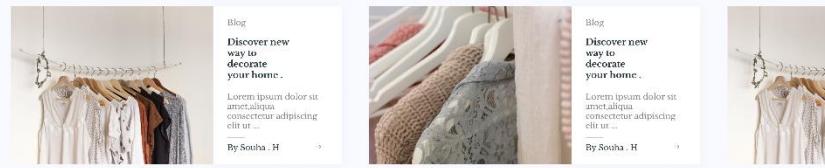
★★★★☆ 4

or adipiscing elit. Duis vel morbi cursus sed sodales molestie proin dicitur gravida. Portitor maecenas sapien feugiat laoreet convallis euismod augue cras eu eget. Risus suspendisse mauris ullamcorper

★★★★☆ 4

or ipsum dolor sit amet, consectetur adipiscing elit. Duis vel morbi cursus sed sodales molestie proin dicitur gravida. Portitor maecenas tincidunt ipsum semper malesuada. In sapien feugiat laoreet convallis eu sed. Sapien et montes, duis tempor euismod augue cras eu eget. Risus suspendisse mauris ullamcorper

Featured Blogs


[View all](#)


About Us

Business Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vel morbi cursus sed sodales molestie proin dicitur gravida. Portitor maecenas tincidunt ipsum semper malesuada. In sapien feugiat laoreet convallis eu sed. Sapien et montes, duis tempor euismod augue cras eu eget. Risus suspendisse mauris ullamcorper felis a, quam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vel morbi cursus sed sodales molestie proin dicitur gravida. Portitor maecenas tincidunt ipsum semper malesuada. In sapien feugiat laoreet convallis eu sed. Sapien et montes, duis tempor euismod augue cras eu eget. Risus suspendisse mauris ullamcorper felis a, quam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vel morbi cursus sed sodales molestie proin dicitur gravida. Portitor maecenas

Contact Information

+91 1234567890
Someting@random.com

[Directions](#)


BAUHINIA

Women

All Women
Skirts
T-Shirts
Tops
Jackets

Men

All Men
Shirts
T-Shirts
Shorts
Jackets

Kids

All Kids
Shirts
T-Shirts
Shorts
Jackets

Shopping

Your cart
your orders
Comparison items
Wishlist
Shipping Details

More links

Blogs
Gift center
Buying guides
New arrivals
Clearance

Stay In Touch

Stay in touch to get special offers, free giveaways and once in a lifetime deals

Enter your email

[Terms & Conditions](#)
[Privacy Policy](#)


Figure 1. 10 Customer - Home Page

Customer – Products Browse Page:

BAUHINIA

- [Home](#)
- [Shop](#) ▾
- [About](#)
- [Blog](#)
- [Contact](#)
- [Update Profile](#)

[Login / Register](#)

Filters [Clear all](#)
Sort By

Filters

[Tag for Brand](#) [Tag for Clothes](#)

[Tag for Clothes Size](#)

Brand

- Sri Lankan Talkies (206)
- Roadster (26)
- Here&Now (706)
- High Star (64)
- Miss Chase (16)
- Voxati (20)
- [+40 more](#)

Price

- Rs 350 to Rs 500 (206)
- Rs 500 to Rs 700 (100)
- Rs 350 to Rs 500 (206)

Color

- Blue (206)
- [+40 more](#)

Discount Range

- 10% and above (26)
- 20% and above (62)
- 30% and above (20)
- 40% and above (106)
- 50% and above (32)
- 60% and above (46)



Women's Hoodies
Brand Name 4.4 ★
Rs. 600 Rs.1000 (30% off)



Men's Jackets
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Women's Party Wear
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Stylish Wear (Unisex)
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Women's Denim Jackets
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Women's Tops
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Women's T-Shirts
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Men's Formal Wear
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Men's T-Shirts
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Womens Skirts
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Sweaters
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Swimwear
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Globex

Women	Men	Kids	Shopping	More links	Stay In Touch
All Women	All Men	All Kids	Your cart	Blogs	Stay in touch to get special offers, free giveaways and once in a lifetime deals
Skirts	Shirts	Shirts	your orders	Gift center	
T- Shirts	T- Shirts	T- Shirts	Compared items	Buying guides	
Tops	Shorts	Shorts	Wishlist	New arrivals	
Jackets	Jackets	Jackets	Shipping Details	Clearance	

[Terms & Conditions](#) [Privacy Policy](#)

Enter your email

Figure 1. 11 Customer – Products Browse Page

Ryan Wickramaratne (COL 00081762)

Unit_30:AD - Application Development

104

Customer – Products Details Page:

Womens Hoodies Jacket (Blue)

Moose

Product ID : 2253421

★★★★★ 4.4 36 Reviews

Rs. 650 Rs.1000 (30% off)

Select Size

Size Chart >

(XS) (S) (M) (L) (XL)

Select Color

(Blue) (Grey) (Red)

Best Offers

Special offer get 25% off T&C

Bank offer get 30% off on Axis Bank Credit card T&C

Wallet offer get 40% cashback via Paytm wallet on first transaction T&C

Special offer get 25% off T&C

Add to cart

Product Details		Specification	Ratings & Reviews																				
Specifications <table border="1"> <tr> <td>Sleeve Length</td> <td>Type</td> </tr> <tr> <td>Long Sleeves</td> <td>Denim Jacket</td> </tr> <tr> <td>Print or Pattern Type</td> <td>Color</td> </tr> <tr> <td>Washed</td> <td>Spread collar</td> </tr> <tr> <td>Length</td> <td>Closure</td> </tr> <tr> <td>Regular</td> <td>Button</td> </tr> <tr> <td>Lining Fabric</td> <td>Number of Pockets</td> </tr> <tr> <td>Unlined</td> <td>4</td> </tr> <tr> <td>Hemline</td> <td>Occasion</td> </tr> <tr> <td>Straight</td> <td>Casual</td> </tr> </table>		Sleeve Length	Type	Long Sleeves	Denim Jacket	Print or Pattern Type	Color	Washed	Spread collar	Length	Closure	Regular	Button	Lining Fabric	Number of Pockets	Unlined	4	Hemline	Occasion	Straight	Casual		
Sleeve Length	Type																						
Long Sleeves	Denim Jacket																						
Print or Pattern Type	Color																						
Washed	Spread collar																						
Length	Closure																						
Regular	Button																						
Lining Fabric	Number of Pockets																						
Unlined	4																						
Hemline	Occasion																						
Straight	Casual																						

Similar Products

Hoodies
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Men's Jackets
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Women's Tops
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Women's Party Wear
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Women's Bottoms
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

BAUHINIA

Women	Men	Kids	Shopping	More links	Stay In Touch
All Women	All Men	All Kids	Your cart	Blogs	Stay in touch to get special offers, free giveaways and once in a lifetime deals
Skirts	Shirts	Shirts	your orders	Gift center	
T- Shirts	T- Shirts	T- Shirts	Compared items	Buying guides	
Tops	Shorts	Shorts	Wishlist	New arrivals	
Jackets	Jackets	Jackets	Shipping Details	Clearance	

Terms & Conditions Privacy Policy

() () () ()

Figure 1. 12 Customer – Products Details Page

Customer – Checkout Page:

The screenshot displays a web-based shopping cart interface. At the top, there's a navigation bar with links for 'Cart', 'Checkout' (which is underlined in blue), and 'Tracking'. A search bar is positioned next to a magnifying glass icon. On the right side of the header, there are icons for a bookmark, a shopping cart, and a user profile labeled 'Ryan'.

The main content area is divided into two main sections: 'Order summary' on the left and 'Complete your order' on the right.

Order summary:

- Womens Black Tshirt:**
Size: 37
Quantity: 1
Total Price: Rs. 1000
- Womens White Skirt:**
Size: 30
Quantity: 1
Total Price: 600

Complete your order:

Personal Details:

First name	Last name
Enter Your First Name...	Enter Your Last Name...
Email	Phone number
Enter Your Email...	Enter Your Phone Number...

Payment Details:

Payment method options shown: VISA, stripe, P, MASTERCARD, G Pay.

Card holder name	Card number
Enter Your Full Name...	Enter Your Card Number...
CVV	Expiration Date
Example: 4567	MMYY

Shipping Address:

Address line 1:
Your Shipping Address...

City	State
Your City	Your State
Landmark	Postal code
Any Landmarks/PostalCodes/Post Office Box	ZIP Code (20296)

Total bill: Rs. 1600/-

Buttons: Cancel (in a grey box) and Complete Purchase (in a blue box).

Figure 1. 13 Customer – Checkout Page

Customer – Shopping Cart Page:

The screenshot shows a web-based shopping cart interface. At the top, there's a navigation bar with links for 'Cart', 'Checkout', and 'Tracking'. A search bar is positioned next to a magnifying glass icon. On the right side of the header, there are icons for a bookmark, a shopping cart, and a user profile labeled 'Ryan'.

The main content area is titled 'My Orders (2 Items)'. It displays two separate order cards.

Order 1: Womens Black Tshirt
Levies
Rs. 1000
Order Placed on : 6 July 2023
Order ID : 3354654654526
Product ID : 225487
[Cancel](#)

Order 2: Womens White Skirt
Moose
Rs. 600
Order Placed on : 1 July 2023
Order ID : 3354654654555
Product ID : 222458
[Cancel](#)

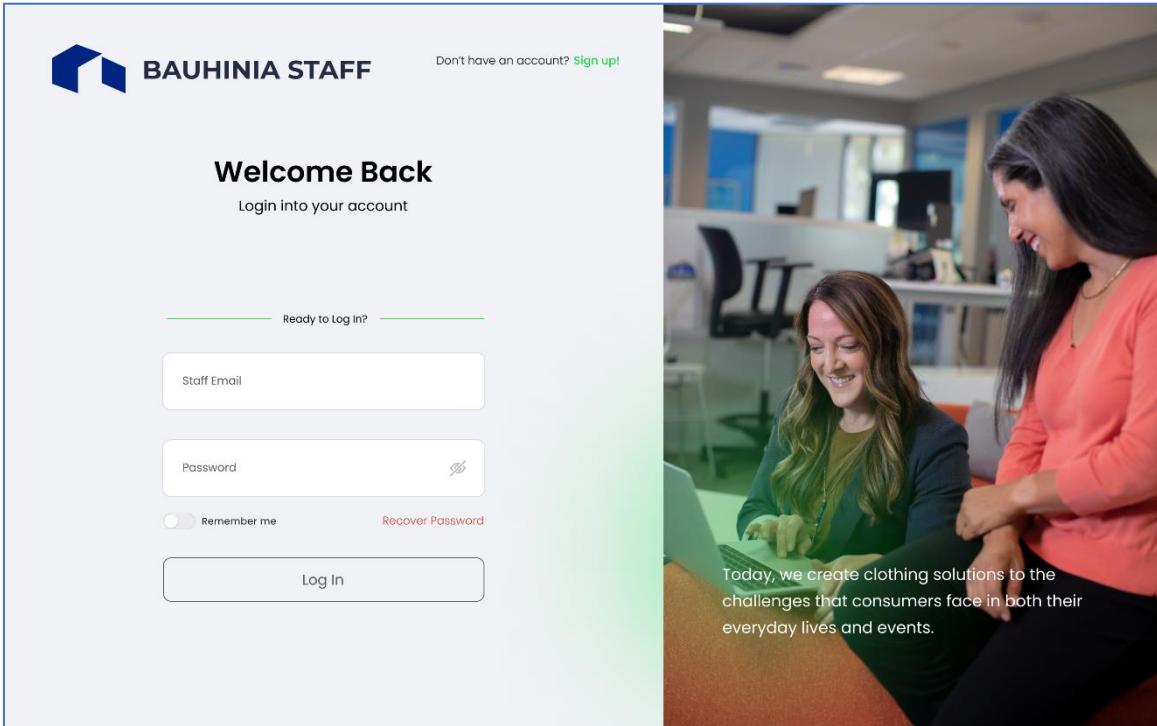
At the bottom of the page, there's a dark footer section containing the BAUHINIA logo, copyright information ('BAUHINIA, 2021 All rights reserved'), and links for 'Shopping' (Your cart, your orders, Compared items, Wishlist, Shipping Details), 'More links' (Blogs, Gift center, Buying guides, New arrivals, Clearance), and 'Social Media' (links to Facebook, Instagram, YouTube, and Twitter).

Figure 1. 14 Customer – Shopping Cart Page

Customer – Order Tracking Page:

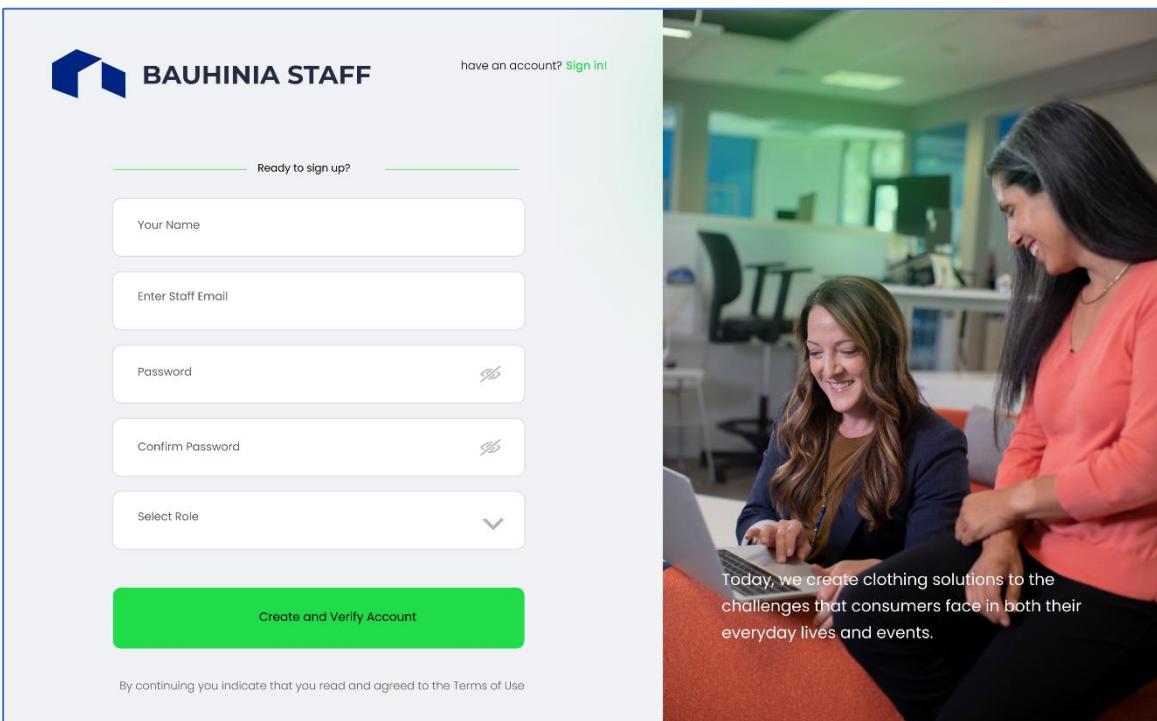
The screenshot displays the customer order tracking interface. At the top, there's a navigation bar with a blue icon, 'Cart', 'Checkout', 'Tracking' (which is underlined), a search bar with placeholder 'Search here', and user icons for 'Ryan'. Below the navigation is a breadcrumb trail: 'Home > Orders > ID 3354654654526'. The main content area shows 'Order ID: 3354654654526' and the date 'Order date: July 6, 2023 | Estimated delivery: August 16, 2024'. A timeline indicates the status: 'Order Confirmed' (green dot at Wed, 6th July), 'Shipped' (grey dot at Fri, 8th July), 'Out For Delivery' (grey dot at Sun, 11th July), and 'Delivered' (grey dot at Mon, 16th August). Two order items are listed: 'Womens Black Tshirt' (Rs. 1000.00) and 'Womens White Skirt'. The left sidebar shows 'Orders to track' with the same two items. At the bottom, it says 'Total Orders 2'.

Figure 1. 15 Customer – Order Tracking Page

Staff – Login Page:

The screenshot shows the 'BAUHINIA STAFF' login page. At the top left is the logo, followed by the text 'BAUHINIA STAFF'. To the right is a link 'Don't have an account? [Sign up!](#)'. Below this is a heading 'Welcome Back' and a sub-instruction 'Login into your account'. A horizontal line labeled 'Ready to Log in?' spans the width of the input fields. There are two input fields: 'Staff Email' and 'Password', each with a clear button. Below them are 'Remember me' and 'Recover Password' buttons. A large green 'Log In' button is at the bottom. To the right of the form is a photograph of two women in an office setting, one pointing at a laptop screen. Below the photo is a testimonial: 'Today, we create clothing solutions to the challenges that consumers face in both their everyday lives and events.'

Figure 1. 16 Staff – Login Page

Staff – Sign-up Page:

The screenshot shows the 'BAUHINIA STAFF' sign-up page. At the top left is the logo, followed by the text 'BAUHINIA STAFF'. To the right is a link 'have an account? [Sign in!](#)'. Below this is a horizontal line labeled 'Ready to sign up?'. There are five input fields: 'Your Name', 'Enter Staff Email', 'Password', 'Confirm Password', and 'Select Role'. A large green 'Create and Verify Account' button is at the bottom. Below the button is a small note: 'By continuing you indicate that you read and agreed to the Terms of Use'. To the right of the form is a photograph of two women in an office setting, one pointing at a laptop screen. Below the photo is a testimonial: 'Today, we create clothing solutions to the challenges that consumers face in both their everyday lives and events.'

Figure 1. 17 Staff – Sign-up Page

Staff – Reports Page:

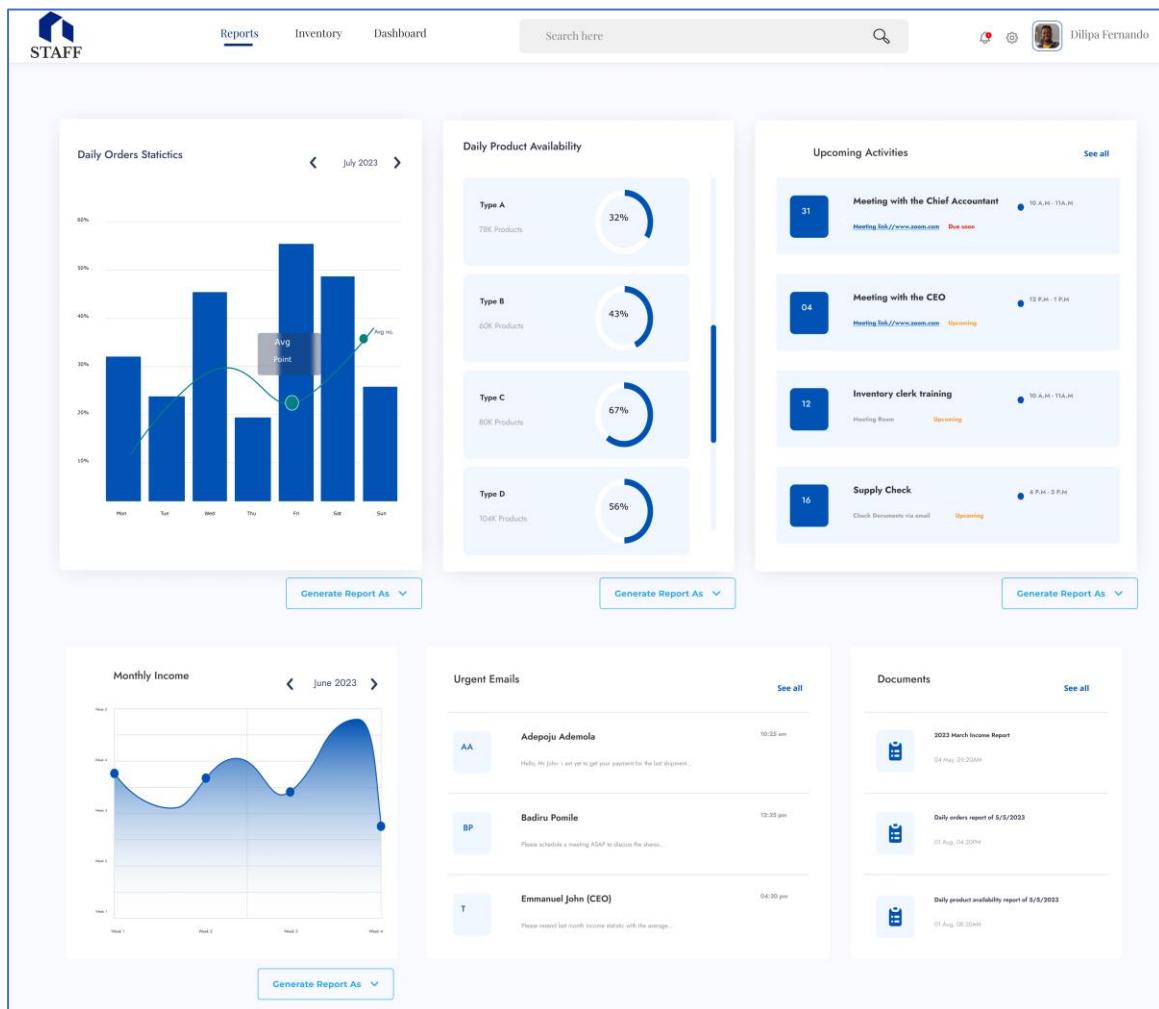


Figure 1. 18 Staff – Reports Page

Staff – Inventory Update Page:

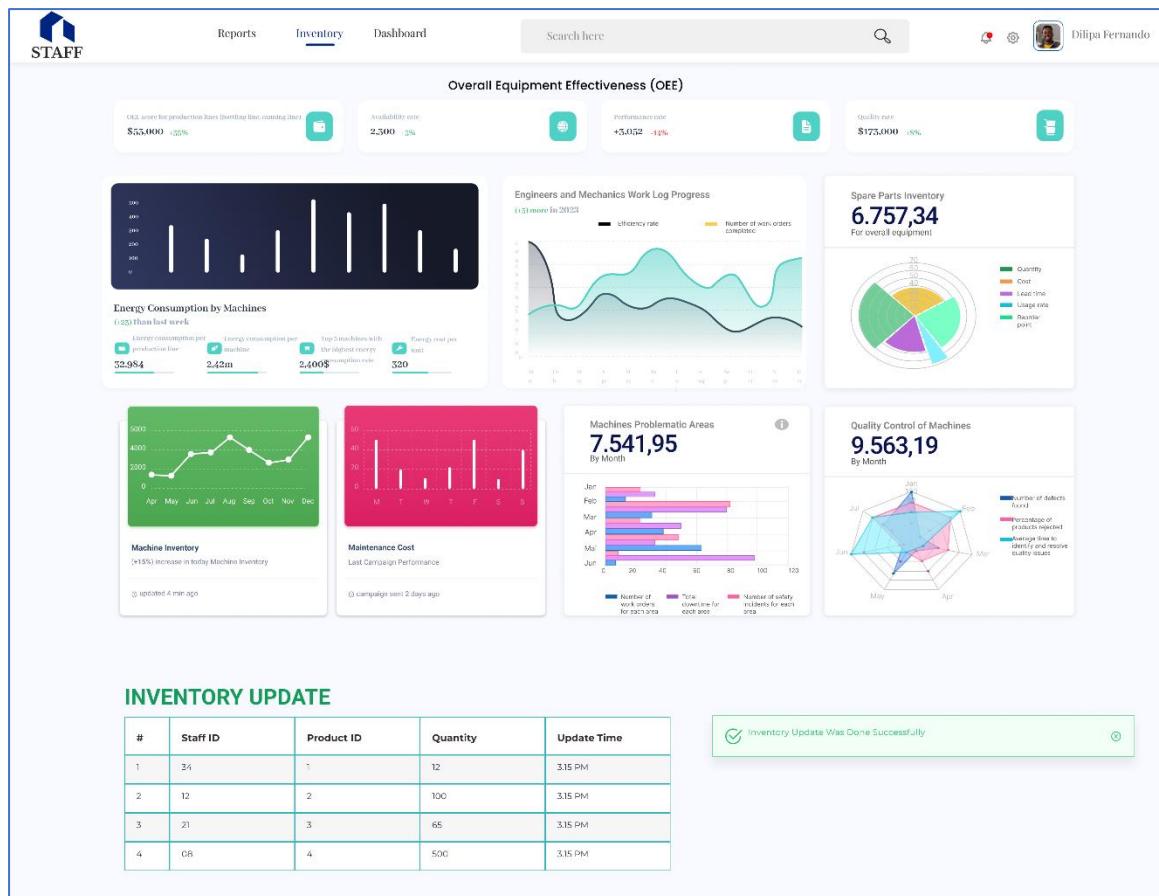


Figure 1. 19 Staff – Inventory Update Page

Staff – Dashboard Page:

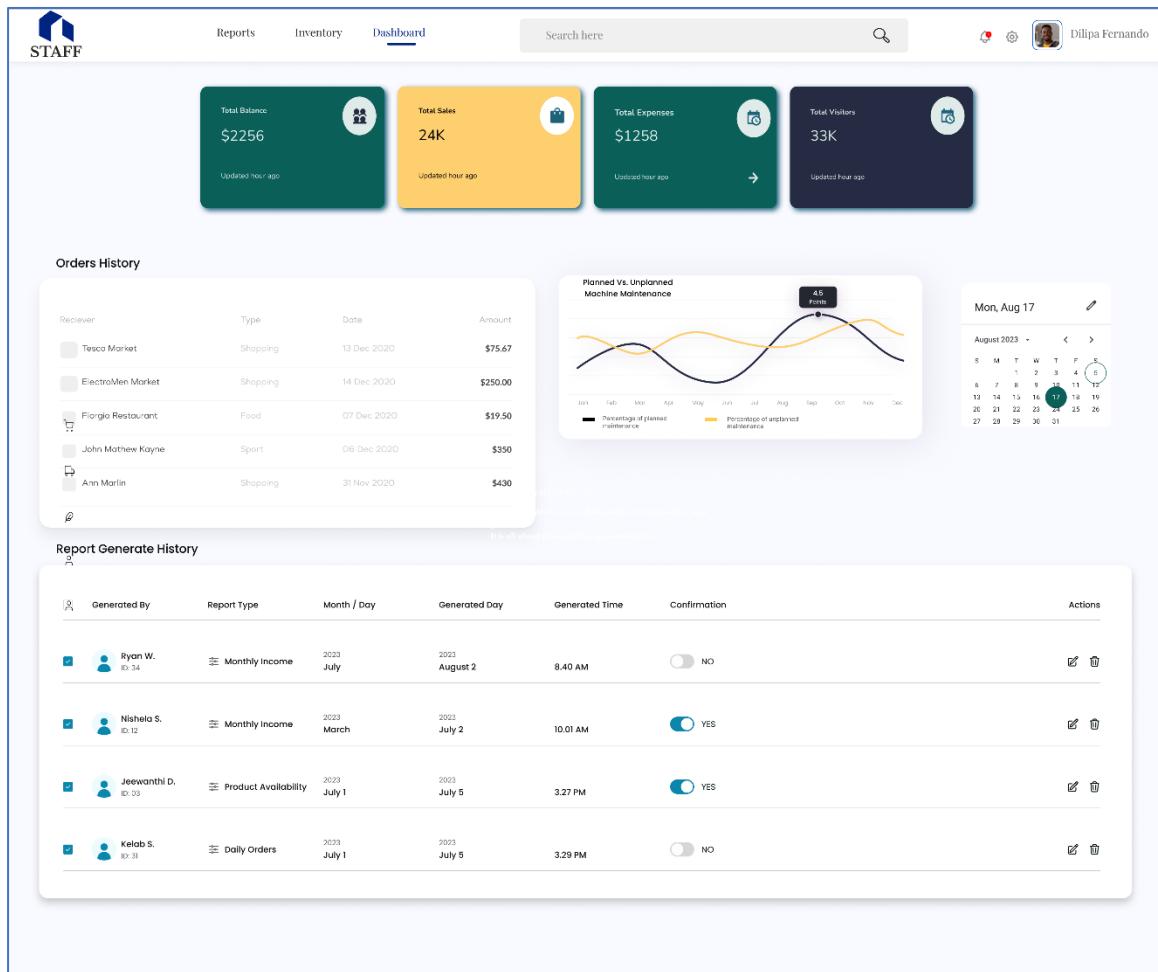


Figure 1. 20 Staff – Dashboard Page

1.5.23 System Design – Logical Database Diagram Design

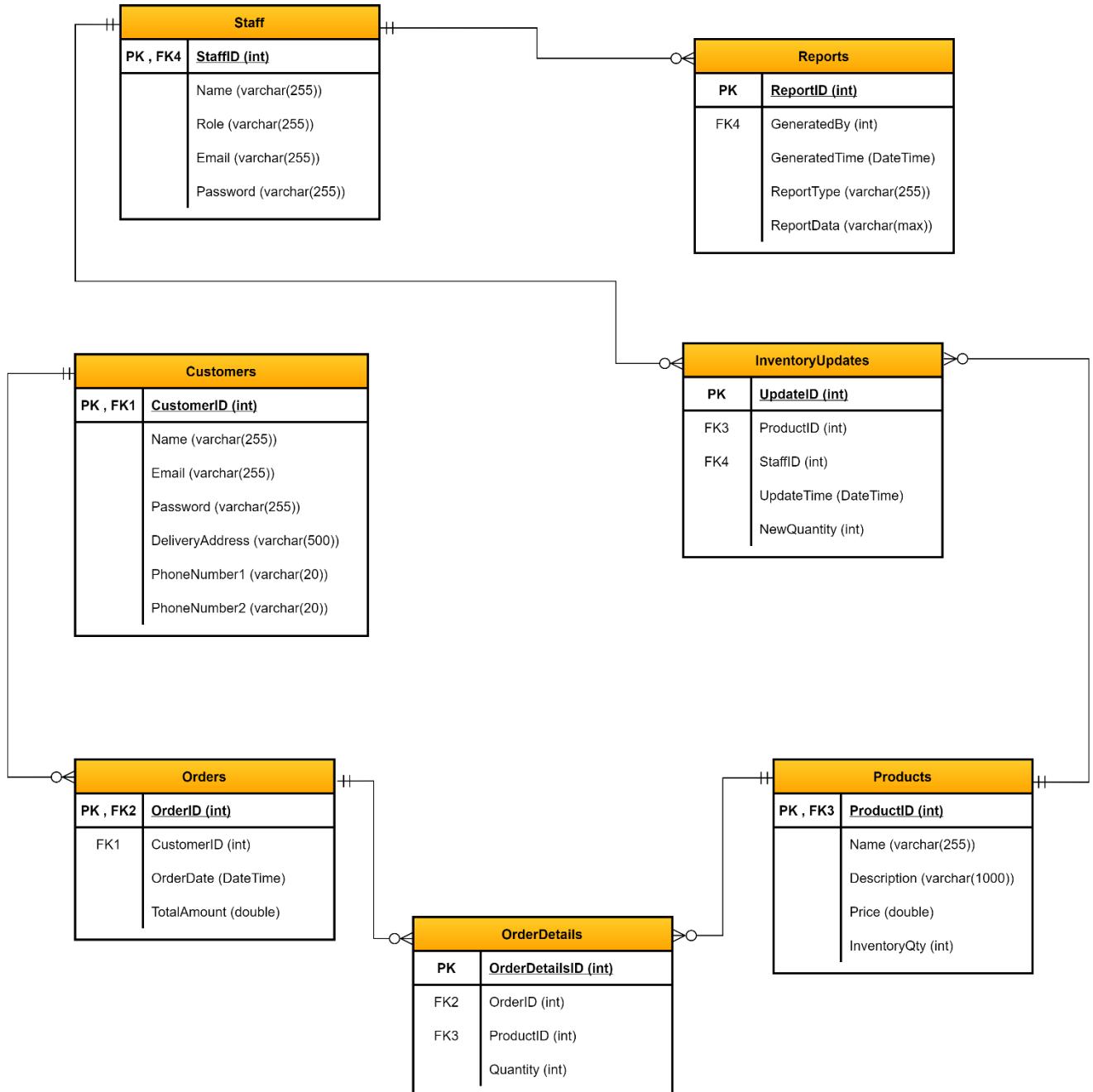


Figure 1. 21 Logical Database Diagram Design

1.5.24 Coding Standards and Conventions

Naming Conventions:

- Use clear and descriptive names for variables, functions, classes, and modules.
- Use appropriate casing. For example, in Python, use snake_case for variable and function names, and PascalCase for class names.
- Constants should be written in all uppercase, with words separated by underscores.
- Avoid using names that are too short or abbreviations that aren't widely accepted.

Commenting and Documentation:

- Provide comments to explain complex or non-obvious parts of the code.
- Every function and class should have a docstring describing what it does, its parameters, and its return value.
- Keep comments and docstrings up-to-date with code changes.
- Use in-line comments sparingly.

Indentation and Spacing:

- Use 4 spaces per indentation level, as per PEP 8 guideline for Python.
- Separate functions and method definitions with two blank lines.
- Class definitions should also be separated by two blank lines.
- Within classes, methods should be separated by one blank line.
- Add a space after a comma in arguments.

Code Organization:

- Each Python file should contain a module, and each module should serve a specific purpose within the application.
- Each class should represent an entity and its associated functionality.

- Avoid making classes or functions too large. If a function or method is becoming too complex, consider breaking it up into smaller, more manageable pieces.

Error Handling:

- Do not leave empty exceptions.
- Use try/except blocks to handle errors and exceptions.
- Log exceptions with appropriate messages.

Code Quality:

- Avoid code duplication. If you're writing the same code in several places, consider moving it to a separate function.
- Use list comprehensions and built-in functions where possible for brevity and performance.
- Avoid using too many nested loops or conditions.

Testing and Debugging:

- Write unit tests to check the functionality of your code.
- Regularly use a linter and a static code analysis tool to check your code for possible errors and ensure that it conforms to the coding standards.

Version Control:

- Use a version control system, such as Git.
- Make small, incremental commits.
- Write clear commit messages.

1.5.25 Coding Implementation Strategy

Understand the Design: First, ensure that all team members thoroughly understand the software design. They should be familiar with the functionality of each part of the system and know how these parts interact with each other.

Divide and Conquer: Assign different system components to different team members or sub-teams based on their expertise. For instance, one team could work on the customer interface, another team could work on the staff interface, and another team could work on the backend logic and database.

Implement Core Functionality First: Start by implementing the core functionality of the system. In the case of the BAUHINIA system, this could include product browsing, cart functionality, and the order process.

Test As You Go: As each piece of functionality is implemented, conduct unit tests to ensure it works as expected. This will make it easier to locate and fix bugs. Use a continuous integration (CI) tool to automate testing and ensure that all new code integrates properly with the existing code.

Progressive Implementation: After the core functionality is in place and tested, we can start adding the less critical features, such as the customer and staff dashboards, report generation, and inventory updates.

Code Reviews: Implement a code review process where every piece of code is reviewed by at least one other developer before being merged. This will help ensure code quality and consistency.

Regular Sync-ups: Hold regular meetings to discuss progress, any issues that have arisen, and next steps. This will ensure everyone is on the same page and can help catch potential problems early.

Version Control: Use a version control system like Git to track changes and make collaboration easier. Make sure to commit early and often, and use clear, descriptive commit messages.

Iterative Development: Aim to have a working, if not complete, version of the system at all times. This allows for regular testing and feedback, and it makes it easier to adjust course if necessary.

Refactoring: Once all features are implemented and the system is working as expected, consider refactoring the code to improve code quality, readability, and performance. This is also a good time to ensure that the code is well-documented.

1.5.26 System Code Implementation

Customer - Login Page:

In views.py (function-based view):

```
1 from django.contrib.auth import authenticate, login
2 from django.shortcuts import render, redirect
3
4 def customer_login(request):
5     if request.method == "POST":
6         email = request.POST['email']
7         password = request.POST['password']
8         user = authenticate(request, username=email, password=password)
9         if user is not None:
10             login(request, user)
11             return redirect('home')
12         else:
13             return render(request, 'login.html', {'error': 'Invalid login credentials.'})
14     else:
15         return render(request, 'login.html')
16
17
```

In the login.html template:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Login</h2>
5 <form method="post">
6     {% csrf_token %}
7     <label for="email">Email:</label><br>
8     <input type="text" id="email" name="email"><br>
9     <label for="password">Password:</label><br>
10    <input type="password" id="password" name="password">
11    <input type="submit" value="Login">
12 </form>
13 {% if error %}
14 <p><strong>{{ error }}</strong></p>
15 {% endif %}
16 {% endblock %}
```

Customer - Sign-up Page:

In views.py (function-based view):

```
1 from django.contrib.auth.models import User
2 from django.contrib.auth import login
3 from django.shortcuts import render, redirect
4
5 def customer_signup(request):
6     if request.method == "POST":
7         username = request.POST['username']
8         password = request.POST['password']
9         email = request.POST['email']
10        user = User.objects.create_user(username=username, email=email, password=password)
11        user.save()
12        login(request, user)
13        return redirect('home')
14    else:
15        return render(request, 'signup.html')
16
```

In the signup.html template:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Sign up</h2>
5 <form method="post">
6     {% csrf_token %}
7     <label for="username">Username:</label><br>
8     <input type="text" id="username" name="username"><br>
9     <label for="email">Email:</label><br>
10    <input type="text" id="email" name="email"><br>
11    <label for="password">Password:</label><br>
12    <input type="password" id="password" name="password">
13    <input type="submit" value="Sign Up">
14 </form>
15 {% endblock %}
16
```

Customer - Home Page:

In views.py (function-based view):

```
1 from django.shortcuts import render
2 from .models import Product
3
4 def home(request):
5     products = Product.objects.all()
6     return render(request, 'home.html', {'products': products})
7
8
```

In home.html:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Welcome to Our Store</h2>
5 <div class="products">
6     {% for product in products %}
7         <div class="product">
8             <h3>{{ product.name }}</h3>
9             <p>{{ product.description }}</p>
10            <p>Price: {{ product.price }}</p>
11        </div>
12        {% empty %}
13        <p>No products available.</p>
14    {% endfor %}
15 </div>
16 {% endblock %}
```

Customer – Products Browse Page:

In views.py (function-based view):

```
1 from django.shortcuts import render
2 from .models import Product
3
4 def browse_products(request):
5     products = Product.objects.all()
6     return render(request, 'browse_products.html', {'products': products})
```

In browse_products.html:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Browse Our Products</h2>
5 <div class="products">
6     {% for product in products %}
7         <div class="product">
8             <h3>{{ product.name }}</h3>
9             <p>{{ product.description }}</p>
10            <p>Price: {{ product.price }}</p>
11            <a href="{% url 'product_detail' product.id %}">View Details</a>
12        </div>
13        {% empty %}
14        <p>No products available.</p>
15        {% endfor %}
16    </div>
17    {% endblock %}
```

Customer – Products Details Page:

In views.py (function-based view):

```
1 from django.shortcuts import render, get_object_or_404
2 from .models import Product
3
4 def product_detail(request, product_id):
5     product = get_object_or_404(Product, pk=product_id)
6     return render(request, 'product_detail.html', {'product': product})
7
```

In product_detail.html:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4     <h2>Product Details</h2>
5     <div class="product">
6         <h3>{{ product.name }}</h3>
7         <p>{{ product.description }}</p>
8         <p>Price: {{ product.price }}</p>
9         <p>Inventory: {{ product.inventory_qty }}</p>
10        <!-- Here you can add a link or a button to add the product to the cart. -->
11        <a href="{% url 'add_to_cart' product.id %}">Add to Cart</a>
12    </div>
13 {% endblock %}
14
```

Customer – Checkout Page:

In views.py (function-based view):

```
1 from django.shortcuts import render, redirect
2 from .models import Order, Customer
3 from .forms import CheckoutForm
4
5 def checkout(request):
6     if request.method == "POST":
7         form = CheckoutForm(request.POST)
8         if form.is_valid():
9             # Assuming the customer is logged in and the customer object is available.
10            customer = Customer.objects.get(user=request.user)
11            order = Order(customer=customer)
12            order.save()
13            return redirect('order_confirmation', order_id=order.id)
14     else:
15         form = CheckoutForm()
16     return render(request, 'checkout.html', {'form': form})
17
18
```

In forms.py:

```
1 from django import forms
2
3 class CheckoutForm(forms.Form):
4     delivery_address = forms.CharField(max_length=200)
5     phone_number = forms.CharField(max_length=15)
6
7
```

In checkout.html:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4     <h2>Checkout</h2>
5     <form method="post">
6         {% csrf_token %}
7         {{ form.as_p }}
8         <button type="submit">Place Order</button>
9     </form>
10    {% endblock %}
11
```

Customer – Shopping Cart Page:

In models.py:

```
1 from django.db import models
2 from django.conf import settings
3 from .models import Product
4
5 class Cart(models.Model):
6     user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
7
8 class CartItem(models.Model):
9     product = models.ForeignKey(Product, on_delete=models.CASCADE)
10    quantity = models.IntegerField(default=1)
11    cart = models.ForeignKey(Cart, related_name='items', on_delete=models.CASCADE)
12
13
```

In forms.py:

```
1 from django import forms
2
3 class CartItemForm(forms.Form):
4     quantity = forms.IntegerField(min_value=1)
5
6
```

In views.py:

```
1 from django.shortcuts import render, get_object_or_404, redirect
2 from .models import Cart, CartItem
3 from .forms import CartItemForm
4
5 def cart(request):
6     user = request.user
7     cart, created = Cart.objects.get_or_create(user=user)
8     if request.method == 'POST':
9         form = CartItemForm(request.POST)
10        if form.is_valid():
11            quantity = form.cleaned_data['quantity']
12            product_id = request.POST.get('product_id')
13            product = get_object_or_404(Product, id=product_id)
14            cart_item, created = CartItem.objects.get_or_create(product=product, cart=cart)
15            cart_item.quantity = quantity
16            cart_item.save()
17        items = CartItem.objects.filter(cart=cart)
18        return render(request, 'cart.html', {'items': items})
19
```

In cart.html:

```
1  {% extends "base_generic.html" %}  
2  
3  {% block content %}  
4      <h2>Shopping Cart</h2>  
5  {% for item in items %}  
6      <div>  
7          <h3>{{ item.product.name }}</h3>  
8          <p>{{ item.product.description }}</p>  
9          <form method="post">  
10             {% csrf_token %}  
11             {{ form.as_p }}  
12             <input type="hidden" name="product_id" value="{{ item.product.id }}>  
13             <button type="submit">Update Quantity</button>  
14         </form>  
15     </div>  
16  {% endfor %}  
17  {% endblock %}
```

Customer – Order Tracking Page:

In models.py file:

```
1 from django.db import models
2 from django.conf import settings
3
4 class Order(models.Model):
5     customer = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
6     order_date = models.DateTimeField(auto_now_add=True)
7     status = models.CharField(max_length=100)
8
9     # Other fields as needed...
10
11 class OrderDetail(models.Model):
12     order = models.ForeignKey(Order, related_name='order_details', on_delete=models.CASCADE)
13     product_name = models.CharField(max_length=200)
14     quantity = models.IntegerField()
15
16     # Other fields as needed...
```

In views.py file:

```
1 from django.shortcuts import render
2 from .models import Order
3
4 def order_tracking(request):
5     orders = Order.objects.filter(customer=request.user).order_by('-order_date')
6     return render(request, 'order_tracking.html', {'orders': orders})
7
```

In order_tracking.html file:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Order Tracking</h2>
5 {% for order in orders %}
6 <div>
7     <h3>Order #{{ order.id }}</h3>
8     <p>Order Date: {{ order.order_date }}</p>
9     <p>Status: {{ order.status }}</p>
10    <h4>Order Details:</h4>
11    <ul>
12        {% for detail in order.order_details.all %}
13            <li>{{ detail.product_name }} (Quantity: {{ detail.quantity }})</li>
14        {% endfor %}
15    </ul>
16 </div>
17 {% endfor %}
18 {% endblock %}
```

Staff – Login Page:

In views.py file:

```
1 from django.contrib.auth import authenticate, login
2 from django.shortcuts import render, redirect
3
4 def staff_login(request):
5     if request.method == 'POST':
6         username = request.POST['username']
7         password = request.POST['password']
8         user = authenticate(request, username=username, password=password)
9         if user is not None and user.is_staff:
10             login(request, user)
11             return redirect('staff_dashboard')
12         else:
13             return render(request, 'staff_login.html', {'error': 'Invalid login credentials'})
14     else:
15         return render(request, 'staff_login.html')
16
```

In staff_login.html file:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4     <h2>Staff Login</h2>
5     <form method="post">
6         {% csrf_token %}
7         <label for="username">Username:</label><br>
8         <input type="text" id="username" name="username"><br>
9         <label for="password">Password:</label><br>
10        <input type="password" id="password" name="password"><br>
11        <input type="submit" value="Login">
12    </form>
13    {% if error %}
14        <p><strong>{{ error }}</strong></p>
15    {% endif %}
16    {% endblock %}
```

Staff – Sign-up Page:

In views.py file:

```
1 from django.contrib.auth.models import User
2 from django.shortcuts import render, redirect
3
4 def staff_signup(request):
5     if request.method == 'POST':
6         username = request.POST['username']
7         password = request.POST['password']
8         email = request.POST['email']
9         if User.objects.filter(username=username).exists():
10             return render(request, 'staff_signup.html', {'error': 'Username already exists'})
11         else:
12             user = User.objects.create_user(username=username, password=password, email=email, is_staff
13                                             =True)
13             user.save()
14             return redirect('staff_login')
15     else:
16         return render(request, 'staff_signup.html')
17
```

In staff_signup.html file:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4     <h2>Staff Sign-up</h2>
5     <form method="post">
6         {% csrf_token %}
7         <label for="username">Username:</label><br>
8         <input type="text" id="username" name="username" required><br>
9         <label for="email">Email:</label><br>
10        <input type="email" id="email" name="email" required><br>
11        <label for="password">Password:</label><br>
12        <input type="password" id="password" name="password" required><br>
13        <input type="submit" value="Sign up">
14    </form>
15    {% if error %}
16        <p><strong>{{ error }}</strong></p>
17    {% endif %}
18    {% endblock %}
```

Staff – Reports Page:

In views.py file:

```
1 from django.shortcuts import render
2 from .models import Reports
3
4 def staff_reports(request):
5     if not request.user.is_staff:
6         return redirect('home') # redirect non-staff users
7     reports = Reports.objects.filter(generated_by=request.user)
8     return render(request, 'staff_reports.html', {'reports': reports})
9
```

In staff_reports.html file:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4 <h2>Your Reports</h2>
5 <ul>
6     {% for report in reports %}
7         <li>
8             <h3>{{ report.report_type }}</h3>
9             <p>Generated on: {{ report.generated_time }}</p>
10            <p>{{ report.report_data }}</p>
11        </li>
12    {% empty %}
13    <p>You have no reports.</p>
14    {% endfor %}
15 </ul>
16 {% endblock %}
```

Staff – Inventory Update Page:

In views.py file:

```
1 from django.shortcuts import render, redirect
2 from .models import InventoryUpdates, Product
3 from .forms import InventoryUpdateForm
4
5 def inventory_update(request):
6     if not request.user.is_staff:
7         return redirect('home') # redirect non-staff users
8     if request.method == 'POST':
9         form = InventoryUpdateForm(request.POST)
10        if form.is_valid():
11            inventory_update = form.save(commit=False)
12            inventory_update.staff = request.user
13            inventory_update.save()
14            return redirect('inventory_update')
15    else:
16        form = InventoryUpdateForm()
17    inventory_updates = InventoryUpdates.objects.filter(staff=request.user)
18    return render(request, 'inventory_update.html', {'form': form, 'inventory_updates': inventory_updates})
```

In forms.py file:

```
1 from django import forms
2 from .models import InventoryUpdates, Product
3
4 class InventoryUpdateForm(forms.ModelForm):
5     class Meta:
6         model = InventoryUpdates
7         fields = ['product', 'new_quantity']
8         widgets = {
9             'product': forms.Select(choices=Product.objects.all()),
10            'new_quantity': forms.NumberInput()
11        }
```

In inventory_update.py file:

```
1  {% extends "base_generic.html" %}  
2  
3  {% block content %}  
4      <h2>Inventory Update</h2>  
5      <form method="POST">  
6          {% csrf_token %}  
7          {{ form.as_p }}  
8          <button type="submit">Update</button>  
9      </form>  
10     <h2>Your Updates</h2>  
11     <ul>  
12         {% for update in inventory_updates %}  
13             <li>  
14                 <h3>{{ update.product.name }}</h3>  
15                 <p>Updated at: {{ update.update_time }}</p>  
16                 <p>New quantity: {{ update.new_quantity }}</p>  
17             </li>  
18         {% empty %}  
19             <p>You have no updates.</p>  
20         {% endfor %}  
21     </ul>  
22  {% endblock %}
```

Staff – Dashboard Page:

In views.py file:

```
1 from django.shortcuts import render, redirect
2 from .models import InventoryUpdates, Reports
3
4 def staff_dashboard(request):
5     if not request.user.is_staff:
6         return redirect('home') # redirect non-staff users
7     inventory_updates = InventoryUpdates.objects.filter(staff=request.user)
8     reports = Reports.objects.filter(generated_by=request.user)
9     return render(request, 'staff_dashboard.html', {'inventory_updates': inventory_updates, 'reports': reports})
10
```

In staff_dashboard.py file:

```
1 {% extends "base_generic.html" %}
2
3 {% block content %}
4     <h2>Staff Dashboard</h2>
5     <h3>Your Inventory Updates</h3>
6     <ul>
7         {% for update in inventory_updates %}
8             <li>
9                 <h4>{{ update.product.name }}</h4>
10                <p>Updated at: {{ update.update_time }}</p>
11                <p>New quantity: {{ update.new_quantity }}</p>
12            </li>
13        {% empty %}
14            <p>You have not made any inventory updates.</p>
15        {% endfor %}
16    </ul>
17    <h3>Your Reports</h3>
18    <ul>
19        {% for report in reports %}
20            <li>
21                <h4>{{ report.report_type }}</h4>
22                <p>Generated at: {{ report.generated_time }}</p>
23                <p>Data: {{ report.report_data }}</p>
24            </li>
25        {% empty %}
26            <p>You have not generated any reports.</p>
27        {% endfor %}
28    </ul>
29 {% endblock %}
```

1.5.27 Test Plan

1. **Introduction:** The goal of this test plan is to ensure that the BAUHINIA system is functioning as expected, and all its components (such as customer functionalities, staff functionalities, and inventory management) are working correctly.

2. **Objectives:** The primary objective is to validate all functionalities of the BAUHINIA system. This includes ensuring that all features work as intended, user interactions are smooth, system performance is optimal, and the system is secure from potential threats.

3. Features to Be Tested:

All functionalities related to Customer and Staff pages:

- Sign-up
- Login
- Home
- Products Browse
- Products Details
- Checkout
- Shopping Cart
- Order Tracking
- Reports
- Inventory Update
- Dashboard

CRUD operations for Customer, Staff, Product, Order, Inventory, and Report entities.

Various workflows such as placing an order, updating inventory, generating reports.

4. Features Not to Be Tested:

Any features or functionalities that are currently out of scope, not yet implemented, or not a part of the current release would not be tested. These should be specified based on the project requirements and status.

5. Risks and Contingencies:

Risks include potential software or hardware failures, resource availability, and schedule adherence. Contingencies will be developing risk mitigation plans, like having backup resources or hardware, and buffering time in the schedule for unexpected delays.

6. Schedule:

The testing process's schedule will be determined based on the project timeline and deadlines. Each phase of testing should have a start and end date, and enough time should be allocated to ensure thoroughness.

7. Responsibilities:

- Test Manager: Responsible for preparing and reviewing the test plan, managing the testing process, and ensuring adherence to the schedule.
- Testers: Responsible for executing the test cases, logging and retesting defects, and preparing test reports.

8. Environmental Needs:

The test environment setup needs to be identical to the production environment. This includes the software, hardware, network configurations, databases, etc. Also, the required test data should be prepared before starting the test execution.

9. Test Techniques:

The system will be subjected to the following types of testing:

- Unit Testing: To test the individual parts of the application (like models, views, forms).
- Integration Testing: To test the interaction between different parts of the application.
- System Testing: To test the application as a whole.
- Usability Testing: To test the user-friendliness of the application.
- Performance Testing: To test the speed and stability of the application under a workload.
- Security Testing: To test the application's ability to protect against threats.

10. Test Cases:

The system will be subjected to various types of testing including Unit Testing, Integration Testing, System Testing, Usability Testing, Performance Testing, and Security Testing. For each of these testing types, specific test cases need to be created. For example:

- Test Case 1: Sign-up functionality for Customers and Staff.
- Test Case 2: Login functionality for Customers and Staff.
- Test Case 3: Product browsing functionality.
- Test Case 4: Product details page functionality.
- Test Case 5: Add to Cart and Modify Cart functionality.
- Test Case 6: Checkout process.
- Test Case 7: Order tracking functionality for Customers.
- Test Case 8: Report generation functionality for Staff.
- Test Case 9: Inventory update functionality for Staff.
- Test Case 10: Dashboard functionality for both Customers and Staff.

11. Pass/Fail Criteria:

Each test case will have its own pass/fail criteria based on the expected result. If the actual result matches the expected result, the test case is considered to have passed, otherwise, it is considered to have failed.

12. Test Deliverables:

Test deliverables include the test plan document, test cases, and test reports which include the details of each test case, their expected and actual results, pass/fail status, and any bugs or issues found.

13. Approvals:

The final test plan needs to be reviewed and approved by the project manager, test manager, and any other relevant stakeholders. Their names, positions, and signatures should be recorded for reference.

1.5.28 Test Cases

Table 1. 13 Test Case 1.1: Customer Sign-up

Test Case 1.1: Customer Sign-up	
Test Scenario: Sign-up functionality for Customers	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Click on the 'Sign up' button.3. Enter valid details in all fields: Name, Email, Password, Delivery Address, and Phone Numbers.4. Click on the 'Create Account' button.	
Expected Result: A new customer account is created. The system should display a confirmation message and redirect the user to the login page.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed	

Table 1. 14 Test Case 1.2: Staff Sign-up

Test Case 1.2: Staff Sign-up	
Test Scenario: Sign-up functionality for Staff	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Click on the 'Staff Sign up' button.3. Enter valid details in all fields: Name, Role, Email, and Password.4. Click on the 'Create Account' button.	
Expected Result: A new staff account is created. The system should display a confirmation message and redirect the user to the login page.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed	

Table 1. 15 Test Case 2.1: Customer Login

Test Case 2.1: Customer Login	
Test Scenario: Login functionality for Customers	
Test Steps:	
1. Open the BAUHINIA system in the web browser. 2. Click on the 'Login' button. 3. Enter valid Email and Password for a customer account. 4. Click on the 'Login' button.	
Expected Result: The system should log in the customer successfully and redirect to the customer's home page.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed	

Table 1. 16 Test Case 2.2: Staff Login

Test Case 2.2: Staff Login	
Test Scenario: Login functionality for Staff	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Click on the 'Staff Login' button.3. Enter valid Email and Password for a staff account.4. Click on the 'Login' button.	
Expected Result: The system should log in the staff successfully and redirect to the staff's dashboard page.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed	

Table 1. 17 Test Case 3: Product Browsing Functionality

Test Case 3: Product Browsing Functionality	
Test Scenario: Verify the product browsing functionality.	
Test Steps: <ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Products' page.4. Browse through the list of products.	
Expected Result: <ol style="list-style-type: none">1. All available products should be listed on the product browsing page.2. Each product should display relevant information such as name, price, and description.3. The products can be sorted and filtered based on different criteria such as product categories, price range, etc. if the feature is provided.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed	

Table 1. 18 Test Case 4: Product Details Page Functionality

Test Case 4: Product Details Page Functionality	
Test Scenario: Verify the product details page functionality.	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Products' page.4. Click on a product to navigate to the product details page.	
Expected Result:	
<ol style="list-style-type: none">1. On clicking a product, the user should be directed to a product details page.2. The product details page should display complete information about the product including name, description, price, and any other details like size, color, etc.3. There should be an option to add the product to the shopping cart.4. If applicable, the product details page should show user reviews and ratings for the product.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed.	

Table 1. 19 Test Case 5: Add to Cart and Modify Cart Functionality

Test Case 5: Add to Cart and Modify Cart Functionality	
Test Scenario: Verify the add to cart and modify cart functionality.	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Products' page.4. Click on a product to navigate to the product details page.5. Click on 'Add to Cart' button.6. Verify that the product is added to the cart and a confirmation message is displayed.7. Navigate to the 'Cart' page.8. Verify the product added is present in the cart with correct details and pricing.9. Change the quantity of the product, and click on 'Update Cart'.10. Verify that the cart is updated correctly with the new quantity and total price.	
Expected Result:	
<ol style="list-style-type: none">1. The product should be successfully added to the cart with a confirmation message displayed to the user.2. The 'Cart' page should accurately reflect the products added, with correct product details, quantity, and pricing.3. Modifying the quantity of a product in the cart should update the cart correctly with the new quantity and total price.	
Actual Result: To be filled after executing the test case	

Status: Pass/Fail (To be filled after executing the test case)
Comments: Any additional notes or issues observed.

Table 1. 20 Test Case 6: Checkout Process

Test Case 6: Checkout Process
Test Scenario: Verify the checkout process functionality.
Test Steps: <ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Products' page.4. Select a product and click on 'Add to Cart' button.5. Navigate to the 'Cart' page and review the order.6. Click on 'Proceed to Checkout' button.7. Fill in all required fields such as shipping address, payment information, etc. (use valid data).8. Click on 'Place Order' button.9. Verify that the order is placed successfully and a confirmation message, along with an order ID, is displayed.10. Verify that an order confirmation email is sent to the registered email address.

Expected Result:

1. User should be able to add a product to the cart and proceed to the checkout page successfully.
2. All required fields on the checkout page should be validated correctly.
3. Upon clicking 'Place Order', the order should be placed successfully and a confirmation message along with an order ID should be displayed.
4. An order confirmation email should be sent to the registered email address with all the correct details.

Actual Result: To be filled after executing the test case

Status: Pass/Fail (To be filled after executing the test case)

Comments: Any additional notes or issues observed.

Table 1. 21 Test Case 7: Order Tracking Functionality for Customers

Test Case 7: Order Tracking Functionality for Customers	
Test Scenario: Verify the order tracking functionality for customers.	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Orders' page.4. Click on the 'Track Order' button/link associated with a particular order.5. Verify that the order tracking page opens and displays the current status of the order.6. Check that the information displayed includes details such as order number, order date, estimated delivery date, and current order status.7. If possible, place a new order and track its status right after the order is placed and at various stages till delivery.	
Expected Result:	
<ol style="list-style-type: none">1. User should be able to navigate to the 'Orders' page and click on 'Track Order' successfully.2. The order tracking page should display the current status of the order.3. The information displayed should be correct and updated according to the current status of the order.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	
Comments: Any additional notes or issues observed.	

Table 1. 22 Test Case 8: Report Generation Functionality for Staff

Test Case 8: Report Generation Functionality for Staff	
Test Scenario: Verify the report generation functionality for staff.	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid staff credentials.3. Navigate to the 'Reports' page.4. Choose the type of report to be generated (Daily Orders, Product Availability, Monthly Income).5. Click on the 'Generate Report' button.6. Verify that the report is generated and displayed on the screen.7. Check the accuracy and completeness of the data in the report.8. If possible, try to generate each type of report multiple times with different date ranges or parameters.	
Expected Result:	
<ol style="list-style-type: none">1. Staff should be able to navigate to the 'Reports' page and select the type of report successfully.2. Upon clicking the 'Generate Report' button, the report should be generated and displayed on the screen.3. The data in the report should be accurate and complete.4. The system should handle multiple report generation requests efficiently.	
Actual Result: To be filled after executing the test case	
Status: Pass/Fail (To be filled after executing the test case)	

Comments: Any additional notes or issues observed.

Table 1. 23 Test Case 9: Inventory Update Functionality for Staff

Test Case 9: Inventory Update Functionality for Staff	
Test Scenario: Verify the inventory update functionality for staff.	
Test Steps:	
<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid staff credentials.3. Navigate to the 'Inventory Update' page.4. Select a product to update the inventory.5. Enter the new inventory quantity in the provided input field.6. Click on the 'Update Inventory' button.7. Verify the inventory update confirmation message.8. Navigate to the product details page to check the updated inventory.9. Repeat the steps for different products and different quantities.	
Expected Result:	
<ol style="list-style-type: none">1. Staff should be able to navigate to the 'Inventory Update' page and select a product to update its inventory successfully.2. After entering the new quantity and clicking on the 'Update Inventory' button, a confirmation message should be displayed.3. The updated inventory quantity should be reflected correctly in the product details page.4. The system should handle different quantities and products without any issues.	
Actual Result: To be filled after executing the test case	

Status: Pass/Fail (To be filled after executing the test case)
Comments: Any additional notes or issues observed.

Table 1. 24 Test Case 10: Dashboard Functionality for both Customers and Staff

Test Case 10: Dashboard Functionality for both Customers and Staff	
Test Scenario:	Verify the dashboard functionality for customers and staff.
Test Steps:	
For Customers:	<ol style="list-style-type: none">1. Open the BAUHINIA system in the web browser.2. Login with valid customer credentials.3. Navigate to the 'Customer Dashboard' page.4. Verify the dashboard elements such as personal profile details, recent orders, and commonly browsed products.5. Click on various elements and verify that they lead to the correct page (e.g., recent order leads to the order tracking page).
For Staff:	<ol style="list-style-type: none">1. Logout from the customer account and login with valid staff credentials.2. Navigate to the 'Staff Dashboard' page.3. Verify the dashboard elements such as profile details, recent inventory updates, and report generation shortcuts.4. Click on various elements and verify that they lead to the correct page (e.g., report generation shortcut leads to the report generation page).

Expected Result:

For Customers:

1. Customers should be able to navigate to the 'Customer Dashboard' page successfully.
2. The dashboard should display the personal profile details, recent orders, and commonly browsed products accurately.
3. Clicking on various elements should lead to the correct pages.

For Staff:

1. Staff should be able to navigate to the 'Staff Dashboard' page successfully.
2. The dashboard should display the profile details, recent inventory updates, and report generation shortcuts accurately.
3. Clicking on various elements should lead to the correct pages.

Actual Result: **To be filled after executing the test case**

Status: **Pass/Fail (To be filled after executing the test case)**

Comments: **Any additional notes or issues observed.**

1.5.29 Test Schedule

Unit Testing (3 weeks)

- Week 1: Unit testing for the customer interface and functionality.
- Week 2: Unit testing for the staff interface and functionality.
- Week 3: Unit testing for the backend logic and database interactions.

Integration Testing (2 weeks)

- Week 4: Integration testing for the customer interface with the backend logic and database.
- Week 5: Integration testing for the staff interface with the backend logic and database.

System Testing (1 week)

- Week 6: Testing the system as a whole, ensuring that all parts work together as expected.

User Acceptance Testing (2 weeks)

- Week 7: User acceptance testing with a small group of end-users, collecting feedback and making any necessary adjustments.
- Week 8: Continue user acceptance testing and finalize the system for release.

Performance and Security Testing (1 week)

- Week 9: Conducting performance testing to ensure the system can handle the expected load, and security testing to ensure data protection.

Regression Testing & Bug Fixes (1 week)

- Week 10: Final round of regression testing after bug fixes and tweaks based on user feedback, ensuring no new issues have been introduced.

Preparation for Launch (1 week)

- Week 11: Preparing the system for launch, including final checks, documentation, and training if necessary.

Post-Launch Monitoring & Maintenance (ongoing)

- Week 12 onwards: After the system is live, continue monitoring for any issues, conduct regular maintenance, and start planning for future updates and enhancements.

1.5.30 System Deployment Strategy

Preparation Phase:

- Verify that all components of the software have been thoroughly tested and validated.
- Prepare the deployment environment, ensuring it has all the necessary hardware and software requirements.
- Document a detailed deployment plan, including a backup plan in case of failure.
- Train the staff who will be handling the deployment.

Staging Phase:

- Deploy the application in a staging environment that closely mirrors the production environment.
- Perform final integration, load, and stress testing to verify the performance and stability of the system.

Production Deployment:

- Once confident that the system is stable, begin deployment in the production environment.
- This could be done during a low-usage period to minimize disruptions to users.
- Use a phased approach, such as a canary or blue/green deployment strategy, to gradually release the system to all users and minimize risks.

Monitoring and Validation:

- Monitor the system closely for any issues during and after the deployment.
- Validate that all functions are working correctly in the production environment.
- Be ready to rollback the deployment if serious issues are detected.

Post-Deployment:

- Conduct a post-deployment review to learn from the process and make improvements for the future.
- Begin regular monitoring and maintenance of the system.
- Plan for regular updates and enhancements based on user feedback and changes in business needs.

1.5.31 System Maintenance and Support Plan

1. Regular System Monitoring and Maintenance:

The system needs to be regularly monitored to ensure optimal performance and availability.

This includes tasks such as:

- Monitoring system performance and usage patterns.
- Regularly checking for security vulnerabilities and updates.
- Regular database backups and optimizing database performance.
- Monitoring error logs and resolving issues.

2. Customer Support:

Dedicated support channels will be established to address any concerns or problems that users face. This includes:

- A helpdesk or support email for customers to report issues.
- A FAQ section or a knowledge base where customers can find solutions to common problems.
- Regular communication of system updates, maintenance periods, or known issues.

3. Software Updates and Enhancements:

The system will need periodic updates and enhancements based on customer feedback and changes in business needs. This includes:

- Regularly releasing patches to fix bugs and vulnerabilities.
- Planning and implementing feature enhancements.
- Regularly reviewing and updating the software documentation.

4. Training and Development:

Ongoing training for both staff and users will be essential to ensure that they are well-versed with the system features and updates. This includes:

- Training new users and staff members on how to use the system.
- Regular training sessions for existing staff members whenever a new feature is added or an existing feature is significantly modified.
- Creating and updating user manuals and training materials.

5. Incident Management:

A well-defined incident management process will be established to handle any issues that arise. This includes:

- A process to record and track incidents.
- Prioritizing incidents based on their severity.
- A team dedicated to resolving these incidents within the agreed service level agreement (SLA) time.

6. Disaster Recovery:

A disaster recovery plan will be in place to handle major incidents like system failure, data breaches, or natural disasters. This includes:

- Regular data backups.

- A plan to restore the system and data in case of failures.
- Regular tests of the disaster recovery plan to ensure its effectiveness.

7. Continuous Improvement:

Continuously gather feedback from users and monitor system usage to identify areas of improvement. This could include:

- Conducting regular user surveys or feedback sessions.
- Analyzing system usage data to identify features that are not being used or are difficult to use.
- Implementing improvements based on feedback and analysis.

1.5.32 Risk Identification

1) Technical Risks:

New Technology: The development team might face difficulties if they are unfamiliar with the Python or Django frameworks used to build the system.

Integration with Existing Systems: There might be difficulties in integrating the new system with BAUHINIA's existing systems. This might lead to data loss or corruption, and could also cause functionality issues.

Performance and Scalability: The system might face performance issues if the number of users or the amount of data exceeds the system's capacity. There might be long loading times, or the system might fail under high load.

Security Vulnerabilities: There might be security vulnerabilities in the system that could be exploited by attackers. This could lead to data breaches or unauthorized access to sensitive information.

2) Project Management Risks:

Project Delays: The project might be delayed due to unforeseen issues during the development process. This could be due to technical issues, changes in project requirements, or team members not completing their tasks on time.

Miscommunication Among Team Members: If there is a lack of clear communication among the development team, it could lead to misunderstandings about project requirements or deadlines. This could cause mistakes and delays.

Scope Creep: There is a risk that new features or requirements could be added to the project after the initial planning phase. This could increase the project's complexity and lead to delays and cost overruns.

Insufficient Resources: There might not be enough resources (e.g., development team members, hardware, software licenses) to complete the project on time and within budget.

3) Financial Risks:

Budget Overruns: The development costs might exceed the allocated budget due to unforeseen expenses, such as the need for additional resources, overtime costs, or additional costs due to project delays or changes in scope.

Unexpected Operational Costs: Post-development costs such as system maintenance, troubleshooting, upgrades, and training could be higher than expected. This could impact the project's overall profitability.

Return on Investment (ROI): If the new system doesn't meet its objectives or isn't adopted by the users as anticipated, the desired ROI may not be achieved.

4) Operational Risks:

System Downtime: The system could experience unexpected downtime due to technical issues, potentially impacting BAUHINIA's ability to serve its customers and staff effectively.

Performance Issues: The system may not perform as expected under real-world conditions. This could result in slow load times, errors, or data loss.

Maintaining and Upgrading the System: It might be difficult to maintain the system or perform necessary upgrades. This could lead to longer downtimes or increased costs.

User Adoption: Staff and customers might have difficulty adopting the new system, which could affect productivity and customer satisfaction.

Training Requirements: Training staff to use the new system might be more challenging or time-consuming than anticipated.

5) Legal and Compliance Risks:

Data Protection Non-Compliance: The system must comply with data protection laws and regulations, such as GDPR (if dealing with European customers) or other regional or

national data protection laws. Non-compliance could result in legal penalties and reputational damage.

Intellectual Property Issues: If any component of the software infringes on the intellectual property rights of others, it could lead to legal issues.

Software Licensing Non-Compliance: The use of third-party software components or libraries must comply with their respective licenses. Non-compliance can also lead to legal issues.

6) Security Risks:

Data Breach: There's a risk of data breaches where unauthorized individuals might access sensitive customer or business data.

Cyberattacks: The system could be vulnerable to various forms of cyberattacks, such as hacking, denial of service (DoS) attacks, or malware.

Unauthorized Access: Insufficient access controls could allow unauthorized individuals to access or manipulate data or systems.

Insufficient Encryption: If the data isn't adequately encrypted, it could be at risk during storage or transmission.

7) Usability Risks:

Complex User Interface: The interface of the application might not be intuitive or user-friendly, causing difficulty for customers and staff members in using the application.

Mismatch of Features with User Expectations: There might be a gap between what the system provides and what users expect. This could lead to dissatisfaction and reduced usage of the system.

Poor Accessibility: The system may not be designed to cater to the needs of all potential users, including those with disabilities. This could limit its use and may cause legal issues related to discrimination.

Inadequate Mobile Experience: If the system is not optimized for mobile use, this could frustrate users who prefer accessing the system via mobile devices, thus affecting user satisfaction.

8) Vendor and Third-party Risks:

Dependency on External Parties: The project might rely on external vendors for certain components or services. If these vendors fail to deliver as expected, it could impact the project timeline and quality.

Inadequate Vendor Support: If vendors do not provide adequate support for their products or services, this could lead to delays and problems in maintaining or updating the system.

Vendor Solvency: If a key vendor goes out of business or discontinues a critical service, the project may suffer.

Third-party System Integration: There could be issues integrating with third-party systems. For instance, if the system is designed to integrate with an existing customer database or payment gateway, any changes or problems with these third-party systems could impact the BAUHINIA system.

9) Organizational Change Risks:

Resistance to Change: Staff members or customers might resist the transition from the old system to the new one, which could affect the adoption rate of the new system.

Inadequate Training: The staff may not receive sufficient training to handle the new system effectively. This could lead to operational inefficiencies and mistakes.

Change Management: Poorly managed change can lead to confusion, loss of productivity, and a drop in morale. Ensuring clear communication, effective training, and adequate support is important to mitigate this risk.

Overhaul of Business Processes: The new system may require changes in existing business processes. If these changes are not managed correctly, it could lead to disruptions in business operations.

10) Scalability Risks:

Inadequate System Capacity: The system may not be designed to handle a large number of users or data, which could lead to system slowdowns or crashes during peak usage times.

Inability to Scale Up Quickly: If the business grows faster than anticipated, the system may not be able to scale up quickly to meet the increased demand.

Cost of Scaling: The cost of scaling the system (e.g., upgrading hardware, purchasing additional server capacity, or increasing the system's capabilities) may be higher than expected, which could strain the project's budget.

Performance Under Load: As the system scales and handles more data and users, there may be unexpected performance issues that weren't evident during testing with lower loads.

1.5.33 Risk Report Forms

Table 1. 25 RISK REPORT FORM – For Technical Risks

RISK REPORT FORM – For Technical Risks		
Date	9 th June 2023	
Risk description	BAUHINIA may face significant technical difficulties during the transition from a manual to a computerized system. This includes potential challenges in data migration, system integration, and managing the learning curve for both staff and customers.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not strictly date driven. However, it may arise during the transition phase when the new system is being implemented and integrated with existing processes.	
Deliverables impacted	The risk could potentially impact multiple deliverables, including system deployment, staff training, customer adaptation, and even the regular business operations of BAUHINIA during the transition period.	
Initial assessment	Impact : High	Likelihood : Moderate
Suggested risk response	BAUHINIA could engage an experienced IT consultant to oversee the transition process and provide necessary technical guidance. Comprehensive training programs should be implemented to familiarize staff with the new system. User manuals and help guides can be developed for reference. For customers, a robust support system can be established to address their queries and concerns about the new system. This includes a FAQ section on the website, customer service hotlines, and instructional videos.	

	<p>Regular backups of business data should be maintained to prevent loss during the transition. A rollback plan should also be in place in case significant issues arise during the system switch.</p> <p>Gradual rollout of the system can be considered, starting with non-critical operations to ensure smooth transition and minimal disruptions to business.</p>
--	---

Table 1. 26 RISK REPORT FORM – For Project Management Risks

RISK REPORT FORM – For Project Management Risks		
Date	9 th June 2023	
Risk description	BAUHINIA's may encounter management-related risks such as poor coordination among team members, miscommunication, or scope creep, leading to delays in project completion.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	Yes, this risk is date-driven. Poor project management could lead to delays, which would impact the predetermined launch date of the new system.	
Deliverables impacted	This risk could potentially impact all project deliverables due to potential delays, miscommunication, and lack of coordination. It could directly impact the development, testing, and deployment of the new system.	
Initial assessment	Impact : Significant	Likelihood : Probable
Suggested risk response	<p>BAUHINIA should consider hiring an experienced project manager who is familiar with managing similar software development projects.</p> <p>Implementing a clear project plan and a strong communication strategy could help to avoid miscommunication and keep all team members aligned.</p> <p>Regular project status meetings should be held to discuss progress, address any issues, and make necessary adjustments to the plan.</p> <p>A clear scope statement should be defined at the beginning of the project, and any changes to the scope should be strictly managed to avoid scope creep.</p> <p>Utilize project management tools for better coordination, timeline management, and communication among team members.</p>	

Table 1. 27 RISK REPORT FORM – For Financial Risks

RISK REPORT FORM – For Financial Risks		
Date	9 th June 2023	
Risk description	BAUHINIA's transition to a digital solution may encounter financial risks. These could arise from unforeseen development costs, overruns on the budget, or inadequate returns on the investment if the system does not meet its objectives or is not as widely adopted as expected.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not strictly date driven. However, it could become significant during any phase of the project if costs exceed the budget or if the expected returns on the investment are not realized post-implementation.	
Deliverables impacted	This risk could potentially affect the entire project as it involves the financial sustainability of the project. Any financial instability could lead to compromises in the quality of the system or delays in the project timeline.	
Initial assessment	Impact : Moderate	Likelihood : Occasional
Suggested risk response	<p>Proper budget planning and allocation should be carried out at the beginning of the project. Regular budget reviews and updates should be made to manage and control costs.</p> <p>For handling potential cost overruns, a contingency fund should be set aside.</p> <p>BAUHINIA should conduct a comprehensive cost-benefit analysis to ensure the project is financially viable.</p> <p>Monitor project progress closely to ensure it aligns with the project budget and timeline.</p> <p>Use of cost management tools can help in effective financial management.</p>	

	Plan for an effective marketing strategy to ensure high adoption rates and good returns on the investment.
--	--

Table 1. 28 RISK REPORT FORM – For Operational Risks

RISK REPORT FORM – For Operational Risks		
Date	9 th June 2023	
Risk description	Operational risks such as system downtime, performance issues, and difficulties in maintaining and upgrading the new system might affect BAUHINIA's transition to a new order tracking system.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Operational risks can arise at any time during the system's lifecycle, including after its successful implementation.	
Deliverables impacted	If operational risks materialize, they could affect the system's performance, user experience, and overall success. In extreme cases, they could even disrupt BAUHINIA's business operations.	
Initial assessment	Impact : Low	Likelihood : Occasional
Suggested risk response	<p>To minimize downtime, BAUHINIA should ensure that the system is thoroughly tested before deployment. Moreover, an IT support team should be on standby to quickly respond to any system issues that may arise.</p> <p>To ensure system performance, regular system maintenance should be scheduled, and any performance issues should be promptly addressed.</p> <p>For maintenance and upgrades, BAUHINIA should consider setting up a contract with the system developer or a third-party service provider. This would ensure that</p>	

	<p>the system stays up-to-date and compatible with future business needs and technologies.</p> <p>BAUHINIA should develop a disaster recovery plan to address potential crises, such as system failures. The plan should outline procedures for data recovery, system restoration, and business continuity.</p>
--	---

Table 1. 29 RISK REPORT FORM – For Legal and Compliance Risks

RISK REPORT FORM – For Legal and Compliance Risks		
Date	10 th June 2023	
Risk description	Legal and compliance risks, such as non-compliance with data protection laws and regulations, can pose a significant risk to the successful completion and operation of BAUHINIA's new order tracking system.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Legal and compliance risks can arise at any time during the system's lifecycle. They can arise from changes in laws and regulations or from inadvertent non-compliance.	
Deliverables impacted	The risk could potentially impact all aspects of the system that involve data handling and user privacy. This includes, but is not limited to, customer registration, product checkout, and report generation.	
Initial assessment	Impact : Catastrophic	Likelihood : Frequent
Suggested risk response	<p>BAUHINIA should conduct a thorough legal review of the proposed system to ensure compliance with all applicable laws and regulations, particularly those related to data protection and privacy.</p> <p>Implement strong data encryption and security measures to protect user data.</p>	

	<p>Regularly update terms of service and privacy policies to reflect changes in law or system practices.</p> <p>Provide necessary training to staff on handling and protecting user data.</p> <p>Consider obtaining professional legal advice, especially when the system undergoes significant changes or when laws and regulations change.</p> <p>Conduct regular audits of system operations and practices to ensure ongoing compliance.</p>
--	---

Table 1. 30 RISK REPORT FORM – For Security Risks

RISK REPORT FORM – For Security Risks		
Date	10 th June 2023	
Risk description	BAUHINIA's new online order tracking system may be susceptible to various security threats such as unauthorized access, data breaches, and cyber attacks. This could compromise customer information and sensitive business data.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk is not date driven. Security risks can occur at any time, especially if appropriate security measures are not in place.	
Deliverables impacted	This risk can affect the confidentiality, integrity, and availability of the online order tracking system. It can also damage BAUHINIA's reputation, customer trust, and could have legal implications due to non-compliance with data protection regulations.	
Initial assessment	Impact : Catastrophic	Likelihood : Probable
Suggested risk response	Ensure the application of standard security practices during the software development process, including secure coding practices and regular security audits.	

	<p>Implement robust authentication and authorization mechanisms to prevent unauthorized access.</p> <p>Use secure communication protocols to protect data during transmission.</p> <p>Regularly update and patch the system to address security vulnerabilities.</p> <p>Regularly back up data and ensure it can be restored quickly in case of a data breach.</p> <p>Educate staff about security best practices, such as recognizing phishing attempts, using strong passwords, and reporting suspicious activities.</p>
--	--

Table 1. 31 RISK REPORT FORM – For Usability Risks

RISK REPORT FORM – For Usability Risks		
Date	10 th June 2023	
Risk description	There's a risk that the new system developed for BAUHINIA may not meet the usability expectations of the users, including both the employees and customers. This could result in low adoption rates, inefficiencies, and customer dissatisfaction.	
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	No, this risk isn't date driven. However, it could surface anytime during the system's operational phase, particularly when users start interacting with the system.	
Deliverables impacted	This risk could potentially affect a range of deliverables, most notably, user satisfaction, system adoption, operational efficiency, and customer loyalty.	
Initial assessment	Impact : Negligible	Likelihood : Remote

Suggested risk response	<p>Prioritize user-centric design in the development of the system, incorporating principles of user experience (UX) and user interface (UI) design.</p> <p>Conduct usability testing with a diverse group of potential users (including both staff and customers) to gain feedback and make improvements before the system is fully implemented.</p> <p>Establish a feedback mechanism, such as a user survey or a suggestion box, for continuous improvement post-implementation.</p> <p>Develop comprehensive user guides, conduct training sessions for staff, and provide easily accessible customer support to facilitate the transition to the new system.</p> <p>Include options for personalization or customization within the system to cater to individual user preferences. This could enhance user acceptance and satisfaction.</p>
-------------------------	---

Table 1. 32 RISK REPORT FORM – For Vendor and Third-party Risks

RISK REPORT FORM – For Vendor and Third-party Risks	
Date	10 th June 2023
Risk description	BAUHINIA may face potential risks related to reliance on external software vendors or service providers. This could involve potential issues such as unavailability of service, poor quality, or non-compliance with agreed terms and conditions, which could impact the success of the application.
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	Yes, this risk could potentially arise during key dates or events such as the system deployment or during system upgrades and maintenance activities. The risk could also become prominent when there is a need for immediate support or service from the vendor.
Deliverables impacted	The risk could potentially impact the overall system quality, system uptime, system maintenance and support, and even the project timeline if there are significant delays or issues from the vendor side.

Initial assessment	Impact : Moderate	Likelihood : Remote
Suggested risk response	<p>BAUHINIA should conduct thorough due diligence before selecting vendors or service providers, including checking their past performance, reputation, financial stability, and compliance with industry standards.</p> <p>Clear and comprehensive contracts should be established with vendors, outlining the expected level of service, penalty clauses for non-compliance, and exit clauses.</p> <p>Regular performance review meetings should be conducted with vendors to ensure they are meeting the agreed standards. Any issues should be addressed promptly.</p> <p>BAUHINIA should have a contingency plan in place in case of significant vendor-related issues, such as having alternate vendors identified or being prepared to bring certain activities in-house if necessary.</p>	

Table 1. 33 RISK REPORT FORM – For Organizational Change Risks

RISK REPORT FORM – For Organizational Change Risks	
Date	10 th June 2023
Risk description	BAUHINIA may encounter resistance to organizational change when implementing the new online ordering system. Both employees and customers might struggle to adapt to the new processes, impacting the adoption rate and overall effectiveness of the system.
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	This risk is not strictly date driven, but it could become more pronounced during the initial rollout and first few months of the new system's operation.
Deliverables impacted	The risk could potentially affect several deliverables, including staff training, customer satisfaction, user adoption rates, and the efficiency of business operations during the transition period.

Initial assessment	Impact : Significant	Likelihood : Probable
Suggested risk response	<p>BAUHINIA should plan for comprehensive training and onboarding sessions for staff members to understand and familiarize themselves with the new system. This will include hands-on practice, demos, and user manuals.</p> <p>A transition period during which both the old and new systems run in parallel could be beneficial. This allows users to adjust to the new system gradually without the pressure of immediate full adoption.</p> <p>BAUHINIA should prepare for customer support and assistance in the initial stages. This could include creating a robust FAQ section on the website, chatbot assistance, customer service hotlines, and instructional videos.</p> <p>Gathering feedback from both employees and customers regularly can help identify pain points and make necessary adjustments to the system or its implementation process. This promotes a sense of inclusion in the change process, potentially easing resistance.</p>	

Table 1. 34 RISK REPORT FORM – For Scalability Risks

RISK REPORT FORM – For Scalability Risks	
Date	10 th June 2023
Risk description	As BAUHINIA's business continues to grow, the new system may face scalability issues, unable to handle the increased user load, transaction volumes, and data storage requirements.
Is the risk date driven? i.e., if the risk was to occur would be on or related to a date or event?	This risk is not date driven, but is instead dependent on the rate of BAUHINIA's business growth, which can fluctuate over time.
Deliverables impacted	This risk could impact system performance, user satisfaction, and the overall operational efficiency of BAUHINIA. It could potentially affect key deliverables

	such as system responsiveness, transaction processing speed, and data storage and retrieval efficiency.	
Initial assessment	Impact : Low	Likelihood : Probable
Suggested risk response	<p>During the system design phase, consider scalability as a primary requirement. The system architecture should be flexible and adaptable, allowing for scaling up or down as per demand.</p> <p>Regularly monitor system performance and user load. Use these insights to foresee potential scalability issues and take proactive measures.</p> <p>Explore cloud solutions, which offer scalability as a feature. It allows resources to be added or removed as required, thus providing flexibility.</p> <p>Perform stress testing and load testing. This helps identify the system's breaking point and provides insights into how it would behave under peak loads.</p> <p>Consider implementing a microservices architecture, where different functionalities of the system are divided into small, independent services. This allows specific services to be scaled up or down based on demand.</p>	

1.5.34 Risk Mitigation Plan

1) Risk Mitigation Plan for Technical Risks:

Technical risks include challenges related to unfamiliar technologies or tools, integration issues with existing systems, and unforeseen technical problems during the development process.

Risk Mitigation Plan:

- Technology Evaluation and Selection: Evaluate multiple technologies and tools before selecting them for the project. Consider the team's familiarity and expertise with the technology, community support, and future maintenance aspects.
- Training and Knowledge Sharing: Arrange training sessions and workshops for the team members to get accustomed to any new technology or tool being used. Create an environment that encourages knowledge sharing among team members.
- Code Reviews and Testing: Implement strict code review procedures and automated testing to identify potential issues early in the development cycle. This will allow you to catch bugs and other technical issues before they become significant problems.
- Prototype Development: Develop prototypes for critical parts of the system, which allows for testing the feasibility and performance of the design. This can also help identify any technical issues early in the process.
- System Integration Planning: Develop a robust system integration plan. Ensure that the new system is compatible with the existing systems to prevent integration issues later.
- Technical Documentation: Maintain thorough technical documentation, including architecture diagrams, database schemas, and code comments. This will ensure that the information about the system is accessible, which can be crucial in troubleshooting technical issues.

- Utilizing Experienced Tech Leads: Leverage the experience of technical leads or senior developers in decision-making processes and in guiding the development team.

2) Risk Mitigation Plan for Project Management Risks:

Project Management Risks relate to the coordination, planning, and management of the software development process. Risks could include delays due to poor planning, miscommunication among the team, or scope creep.

Risk Mitigation Plan:

- Project Planning: Create a detailed project plan with clear timelines and milestones. This plan should be updated regularly to reflect changes and progress. Utilize project management software to track and manage tasks efficiently.
- Communication Strategy: Establish a clear communication strategy within the team and with stakeholders. Regular meetings and status updates should be scheduled to ensure everyone is aligned and informed.
- Scope Management: Clearly define the scope of the project at the outset. Any changes to the scope should go through a formal change management process that assesses the impact on time, cost, and resources.
- Risk Management Strategy: Have a risk management strategy in place that identifies potential risks, assesses their impact, and plans for their mitigation. Regularly review and update this plan throughout the project.
- Resource Allocation: Ensure appropriate allocation of resources for all tasks. This should be continuously monitored and adjusted as needed. Overloading team members can lead to burnout and mistakes.
- Dependency Management: Map out dependencies between tasks and manage them effectively. Delays in one task can impact others, so it's essential to manage these dependencies proactively.
- Using Agile Methodologies: Use Agile methodologies, such as Scrum or Kanban, to enhance team collaboration, facilitate flexibility, and deliver value incrementally.

- Training and Skill Development: Provide training and skill development opportunities for team members. This will ensure that they have the skills needed to effectively contribute to the project.

3) Risk Mitigation Plan for Financial Risks:

Financial Risks pertain to the financial aspects of the project, such as budget overruns due to unforeseen costs, or failure to achieve the desired return on investment if the system does not meet its objectives or is not adopted by users as expected.

Risk Mitigation Plan:

- Budgeting and Financial Planning: Develop a detailed financial plan and budget at the onset of the project. Regularly review and update this budget as the project progresses.
- Cost Management: Implement strict cost management measures. Monitor and control all project-related costs to ensure they remain within the budget. Any potential cost overruns should be flagged and addressed promptly.
- Financial Contingency Plan: Have a financial contingency plan in place to deal with unexpected costs. This should include a buffer or reserve fund that can be used in case of unforeseen expenses.
- Regular Financial Reporting: Regular financial reports should be generated and reviewed to track spending and identify any issues as early as possible. These reports should include forecasts to highlight any potential future financial risks.
- ROI Analysis: Perform a return on investment (ROI) analysis to ensure the project is financially viable. The project should not only cover its costs but also deliver a reasonable return on the investment made.
- User Adoption Strategies: Implement strategies to promote user adoption of the new system, as this will directly impact the ROI. This could involve user training, support, and regular communication about the benefits and features of the new system.

- Contract Management: Carefully manage contracts with vendors and third parties to ensure they deliver value for money. Regularly review these contracts and renegotiate if necessary.
- Efficient Resource Utilization: Utilize resources efficiently to minimize waste. This includes both human resources and material resources.

4) Risk Mitigation Plan for Operational Risks:

Operational risks involve potential challenges that could arise during the system's operational life. For instance, there could be risks related to system downtime, performance issues, or difficulties in maintaining and upgrading the system.

Risk Mitigation Plan:

- Robust System Design: The system design should be robust and flexible enough to handle various operational challenges. This includes designing for redundancy to minimize the impact of system downtime.
- Performance Monitoring: Regularly monitor the system's performance to identify and address any issues promptly. This includes monitoring load times, server response times, and any other key performance indicators (KPIs) that are relevant to your system.
- Regular Maintenance and Upgrades: Schedule regular maintenance and upgrades to keep the system running smoothly and to ensure it remains up-to-date with the latest technological advancements.
- Disaster Recovery Plan: Have a disaster recovery plan in place to minimize the impact of any catastrophic events like a major system failure. This plan should outline the steps to be taken to restore the system to normal operation as quickly as possible.
- User Training: Train users on how to properly use and maintain the system. This will help prevent operational issues caused by user errors.
- Documentation: Maintain thorough documentation of the system and its operation. This can help in troubleshooting and fixing any operational issues that arise.

- Customer Support: Provide reliable customer support to assist users with any operational issues they encounter. This will not only help in resolving issues quickly but also in maintaining user satisfaction and trust in the system.
- Security Measures: Implement strong security measures to protect the system and its data from cyber threats, which could disrupt its operation.

5) Risk Mitigation Plan for Legal and Compliance Risks:

Legal and compliance risks refer to potential legal issues or non-compliance with regulations, such as data protection laws. For instance, if the system does not adequately protect customer data, BAUHINIA could face legal penalties, as well as damage to its reputation.

Risk Mitigation Plan:

- Regulatory Compliance: Be aware of and adhere to all relevant regulations and laws in all areas where the system will operate. This includes local, state, and federal laws as well as industry-specific regulations.
- Legal Counsel: Consult with legal professionals on a regular basis to ensure that the system remains compliant with all legal requirements and to identify any potential legal risks in a timely manner.
- Data Protection: Implement robust data protection measures to ensure the privacy and security of user data. This includes encryption, secure data storage, and secure data transmission.
- Privacy Policy and Terms of Service: Ensure that the system's privacy policy and terms of service are clearly defined and communicated to users. These should be easily accessible and should comply with all relevant laws and regulations.
- User Consent: Ensure that user consent is obtained where required, such as before collecting or processing user data. Users should also be able to easily withdraw their consent if they choose.
- Compliance Training: Provide training to all team members on relevant laws and regulations to ensure understanding and compliance throughout the organization.

- Audit and Review: Regularly review and audit the system for compliance. This can help identify any potential issues before they become significant legal risks.

6) Risk Mitigation Plan for Security Risks:

Security risks refer to potential vulnerabilities and threats that could compromise the system's security. For instance, there could be risks of data breaches, unauthorized access, or other forms of cyberattacks.

Risk Mitigation Plan:

- Secure Architecture: Develop the system using secure coding practices and a security-focused architecture. This includes using input validation, output encoding, and other measures to prevent common web vulnerabilities such as SQL injection and cross-site scripting.
- Encryption: Use encryption for data at rest and in transit. All sensitive data, such as user credentials and personal information, should be encrypted.
- Authentication and Authorization: Implement strong user authentication and access control mechanisms. This can include multi-factor authentication, strong password policies, and role-based access control.
- Secure Deployment: Ensure that the production environment is secure. This includes hardening servers, using firewalls and intrusion detection systems, and keeping all systems up to date with the latest security patches.
- Penetration Testing: Regularly conduct penetration testing to identify and fix security vulnerabilities. This should be done both during development and in the production environment.
- Incident Response Plan: Have an incident response plan in place to quickly react to any security incidents. This should include steps to identify, contain, eradicate, recover from, and learn from security incidents.
- Security Training: Provide security training to all team members to make them aware of common security threats and best practices for avoiding them.

- Data Backup and Recovery: Regularly back up data and ensure that there are plans in place for data recovery in case of a data loss incident.
- Continuous Monitoring: Continuously monitor the system for any unusual or suspicious activity. This can help detect potential security threats early.

7) Risk Mitigation Plan for Usability Risks:

Usability risks involve the potential for the system to be difficult to use or not meet user expectations. If the system's user interface is not intuitive, or if it does not provide the expected functionalities, users might not adopt the system, leading to a failure in achieving the project's objectives.

Risk Mitigation Plan:

- User-Centered Design: Develop the system with a focus on user-centered design. This involves understanding the needs and expectations of the users and designing the system to meet those needs.
- Usability Testing: Conduct usability testing early and often. This involves testing the system with real users to identify any issues that may affect usability. Feedback from these tests should be used to make improvements.
- Prototyping: Create prototypes of the system or specific features to get user feedback before the full system is developed. This can help identify potential usability issues early.
- Consistent Design: Ensure that the design of the system is consistent. This includes having consistent navigation, colors, fonts, and terminology. Consistency can make the system easier to use and learn.
- Accessibility: Make sure that the system is accessible to all users, including those with disabilities. This could involve following web accessibility standards and guidelines.
- Training: Provide sufficient training and documentation to users to help them understand how to use the system. This could involve creating user guides, FAQs, video tutorials, and other training materials.

- User Feedback: Implement mechanisms for collecting user feedback after the system has been deployed. This feedback can provide valuable insights into potential usability issues and areas for improvement.
- Continuous Improvement: After the system is deployed, continue to make improvements based on user feedback and changing user needs.

8) Risk Mitigation Plan for Vendor and Third-party Risks:

Vendor and third-party risks involve dependencies on external parties, such as software vendors or service providers. There could be risks related to the quality of third-party services, or delays or disruptions due to issues on the vendor's side.

Risk Mitigation Plan:

- Due Diligence: Thoroughly research and assess potential vendors and third parties before entering into any contracts. Look at their track record, financial stability, reputation in the industry, and reviews from previous clients.
- Contracts and SLAs: Define clear contracts and Service Level Agreements (SLAs) with all third parties. These documents should clearly outline expectations, responsibilities, and consequences for not meeting those expectations.
- Contingency Plans: Have contingency plans in place for critical third-party services. This could involve having backup vendors or in-house capabilities to provide the service if needed.
- Regular Communication and Monitoring: Maintain regular communication with vendors and continuously monitor their performance. Early detection of any issues can allow for quicker resolution and reduce the potential impact on the project.
- Security Audits: Conduct regular security audits of third-party services, especially if they handle sensitive data. Make sure they comply with necessary security standards and regulations.
- Vendor Management Process: Establish a vendor management process to oversee and manage relationships with vendors. This process should include regular reviews of vendor performance.

- Diversification: Avoid relying on a single vendor for critical services. Diversification can reduce the impact if one vendor fails to deliver.

9) Risk Mitigation Plan for Organizational Change Risks:

Organizational change risks pertain to the potential challenges associated with introducing a new system within the organization. For instance, there could be resistance from employees who are used to the old system, or difficulties in training employees to use the new system.

Risk Mitigation Plan:

- Change Management Strategy: Develop and implement a comprehensive change management strategy that focuses on communicating the benefits of the new system, addressing employee concerns, and supporting employees during the transition.
- Training Programs: Implement comprehensive training programs to ensure that all users of the system understand how to use it effectively. This training should be tailored to different roles and levels within the organization.
- Communication: Maintain open and transparent communication about the changes. This includes explaining the reasons for the change, the benefits it will bring, and how it will affect individual employees and teams.
- Pilot Testing: Conduct pilot testing with a small group of users to identify potential issues and get feedback before a full-scale rollout. This allows you to make necessary adjustments and improvements before the system is introduced to the entire organization.
- Support and Assistance: Provide continuous support and assistance to employees during and after the system implementation. This can include a dedicated helpdesk, FAQs, user manuals, and one-on-one assistance.
- Feedback Mechanism: Establish a feedback mechanism that allows employees to voice their concerns, suggestions, or issues related to the new system. Regularly review and address this feedback.

- Leadership Engagement: Engage leadership and management in the change process. They can play a key role in promoting the change and addressing employee concerns.
- Gradual Implementation: Consider implementing the new system gradually rather than all at once. This gives employees time to adapt to the new system and reduces the potential for resistance and confusion.

10) Risk Mitigation Plan for Scalability Risks:

Scalability risks relate to the system's ability to scale and handle increased user load or data volume. For instance, if BAUHINIA's business grows rapidly, the system might not be able to handle the increased demand, leading to performance issues or system failures.

Risk Mitigation Plan:

- Scalable Architecture: Design and implement a scalable software architecture from the outset. This may involve using a microservices architecture, which allows different parts of the system to scale independently, or leveraging cloud-based services that can easily scale up or down as needed.
- Performance Testing: Regularly conduct performance testing and load testing to simulate high user load or data volume and identify any bottlenecks or performance issues.
- Monitoring: Implement robust system monitoring to continually track system performance and identify any emerging issues before they become significant problems.
- Proactive Capacity Planning: Based on the business growth projections and the results from performance testing and monitoring, proactively plan for increased capacity. This could involve adding more server capacity, upgrading hardware, or leveraging auto-scaling features in cloud-based environments.
- Optimization: Continually optimize the system to improve performance and efficiency. This could involve tasks like optimizing database queries, improving code efficiency, or implementing caching strategies.

- Data Management Strategy: Have a robust data management strategy in place to manage the increased data volume. This could involve techniques like data archiving, partitioning, or implementing a data warehouse or data lake.
- Redundancy and Failover Plans: Implement redundancy and failover mechanisms to ensure that if one part of the system fails, others can take over to maintain service.

1.5.35 Conclusion

This Software Design Document (SDD) provides a comprehensive view of the design, development, and implementation plans for the BAUHINIA system. It has outlined the various requirements of the different user types including Customers, Inventory Handling Clerk, Production Manager, Chief Accountant, and others. An analysis of the existing system was conducted to identify its shortcomings and opportunities for improvement, leading to the formulation of the proposed system.

In assessing the feasibility of the project, the Operational, Economic, Legal, and Schedule feasibilities were evaluated. The project has been found feasible in all these aspects, confirming the project's viability.

The System Development Methodology selected for this project is Agile, which allows flexibility and iterative progress. The system design section presents an ERD, Class Diagram, Use Case Diagram, Sequence Diagrams, User Interfaces Design, and a Logical Database Design to visually represent the system components and their interactions.

To ensure the maintainability and readability of the system code, Coding Standards and Conventions have been established. An efficient Coding Implementation Strategy has also been set forth, defining the approach for the development phase. The implemented system code will be subject to rigorous testing as per the outlined Test Plan and Cases. A Test Schedule has also been created to ensure that all testing activities are completed timely.

Deployment of the BAUHINIA system will be carried out as per the System Deployment Strategy. The System Maintenance and Support Plan aims to provide continuous support and updates to the system, thereby ensuring its smooth functioning post-deployment.

Risks associated with the system were identified and discussed, ranging from Technical, Project Management, Financial, Operational, Legal and Compliance, Security, Usability, Vendor and Third-party, Organizational Change, to Scalability risks. Risk Report Forms have been completed to provide a detailed understanding of these potential challenges. Moreover, mitigation plans have been proposed to manage and reduce these risks.

In conclusion, the BAUHINIA system has been meticulously planned and designed to offer an efficient and user-friendly solution for the organization's needs. The adoption of the

Agile methodology, detailed system design, well-structured code, thorough testing, and risk management strategies all contribute to the confidence in the successful development and implementation of the BAUHINIA system. With continuous maintenance and support, the system is expected to provide significant value to the organization, its staff, and customers.

1.6 Justification of selected solutions

The choice of software development tools and techniques can greatly impact the successful completion and delivery of a project. These choices should be influenced by the specific needs and characteristics of the project at hand. In the case of BAUHINIA, we have made the following choices:

1.6.1 Agile Development Methodology - Scrum:

The Agile Development Methodology - Scrum is an iterative and incremental project management approach that emphasizes flexibility, collaboration, and customer satisfaction. Scrum is named for a formation in the game of rugby where everyone plays a role in achieving a common goal - and this approach is quite appropriate to describe how the methodology works.

Here's why Scrum was chosen for the BAUHINIA project:

- Iterative and Incremental Development: The Scrum approach breaks the project down into small manageable units called sprints, which typically last 1-4 weeks. Each sprint has a defined scope and a set of deliverables. This allows for frequent inspection and adaptation if necessary, ensuring that the product or solution remains aligned with customer expectations and requirements.
- Collaboration and Communication: Scrum encourages daily communication among team members and stakeholders through daily standup meetings. This allows for issues to be identified and addressed promptly. It also fosters a sense of ownership and collaboration among team members.
- Adaptability: Scrum can accommodate changes in requirements more effectively than traditional methodologies like the Waterfall model. As BAUHINIA is transitioning from a manual ordering system to a fully automated one, it is expected that some requirements will change or new ones will be identified as the project

progresses. Scrum allows for such changes to be integrated into the development process.

- Customer Satisfaction: The regular delivery of functional software at the end of each sprint provides customers with tangible evidence of progress. This enhances customer confidence and satisfaction.
- Risk Reduction: Because working software is delivered after each sprint, risks associated with delivering a large, final product are reduced. Potential issues can be identified early on in the project lifecycle and rectified in subsequent sprints.
- Improved Product Quality: Scrum's emphasis on regular testing and review during development helps to improve the quality of the product. By detecting and fixing defects early, the final product is more likely to meet the desired standards of quality.

Compared to other Agile methods, such as Kanban or Extreme Programming, Scrum provides a good balance of structure and flexibility. While Kanban emphasizes continuous flow and doesn't prescribe specific roles or iterations, Scrum's structured approach can better cater to a project like BAUHINIA where specific goals and deliverables are set for each sprint. Extreme Programming, while also iterative and flexible, places a heavy emphasis on the engineering practices which may not be necessary for the BAUHINIA project.

In conclusion, Scrum's principles and practices are well-aligned with the BAUHINIA project's requirements and context, making it an ideal choice for this endeavor.

1.6.2 UML for System Design:

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system. It's a rich language that can be used to model not just the structure of software systems, but the behavior as well.

Here's why UML was chosen for the BAUHINIA project:

- **UML for Use Case Design:** Use Case diagrams in UML represent the functional requirements of the system and the interaction of the system with its environment. Each use case represents a specific functionality of the system and the actors (users or systems) that interact with it. Use case diagrams provide a wide view of the different types of users of a system and how they interact with it. This makes them an ideal tool to represent the requirements of the BAUHINIA system from the perspective of its different user roles, i.e., Customers, Inventory handling Clerks, Production Managers, Chief Accountants, etc.
- **UML for ERD Design:** Although not part of the UML, Entity Relationship Diagrams (ERDs) serve a complementary role to UML diagrams in database design. ERDs focus on the relationships between different entities in the system like Customers, Orders, Products, and Inventory. It is instrumental in setting up the database schema, which will support the efficient management of data within the system.
- **UML for Class Diagram Design:** Class diagrams are central to object-oriented systems and provide a static structure of the system, representing its classes, their attributes and methods, and relationships among objects. In the context of the BAUHINIA system, class diagrams will be instrumental in defining the structure of different modules, establishing their interconnections, and serving as a blueprint for the coding phase.
- **UML for Sequence Diagram Design:** Sequence diagrams in UML represent the dynamic behavior of a system, demonstrating how objects interact with each other in a particular scenario of a use case. For the BAUHINIA system, sequence

diagrams will provide a clear visualization of how system processes unfold during execution and how different elements interact with each other.

- **UML for User Interface Design:** Although UML does not specifically cater to User Interface design, some elements can be adapted for this purpose, like using activity diagrams to model the flow between screen views or using class diagrams to represent interface controls. However, User Interface design is typically better addressed using other techniques and tools that are specifically designed for it, such as wireframing and prototyping tools.
- **UML for Database Design:** As previously mentioned, while ERDs are commonly used for database design, UML class diagrams can serve as an effective alternative, providing a visual representation of the system's data structure. For the BAUHINIA system, this will ensure efficient data management and facilitate operations like data retrieval, insertion, and modification.

To summarize, the use of UML in the BAUHINIA system design offers a standardized, clear, and coherent way of visualizing and designing the system's components and their interactions. This can greatly enhance the understanding of the system for all stakeholders and facilitate the development process.

1.6.3 Relational Database:

A Relational Database Management System (RDBMS) is a type of database management system that stores data in a structured format, using rows and columns. This makes it possible to identify and access data in relation to another piece of data in the database. The relational model means that the logical data structures—the data tables, views, and indexes—are separate from the physical storage structures. This separation enables administrators to manage physical data storage without affecting access to that data as a logical structure.

For the BAUHINIA system, an RDBMS is an ideal choice for several reasons:

- Structured and Organized Data: The RDBMS model organizes data into tables (or "relations"), which can be easily understood and used by administrators, developers, and users. Each table holds information about a specific entity (such as customers, orders, inventory), with columns representing different attributes of that entity and rows representing individual records.
- Data Consistency and Integrity: An RDBMS has built-in mechanisms to enforce data integrity and consistency. This includes primary and foreign key constraints to ensure relationships between tables are maintained correctly, and data types to ensure that each piece of data is of the appropriate type. For example, if we have an "Orders" table and a "Customers" table, an RDBMS would ensure that each order is linked to a valid customer, preventing data inconsistencies.
- Scalability: RDBMS systems are highly scalable. As BAUHINIA grows and the volume of data increases, an RDBMS can handle larger amounts of data without a significant impact on performance. This scalability extends to concurrent users, ensuring that the system remains responsive and reliable as usage increases.
- Support for Complex Queries: With Structured Query Language (SQL), an RDBMS provides a powerful interface to query and manipulate data. SQL enables complex queries across multiple tables, allowing BAUHINIA to generate detailed reports and perform comprehensive data analysis.

- Security: RDBMS systems offer robust security mechanisms. Administrators can grant different access privileges to different users, ensuring that individuals only have access to the data they need. This will be important for BAUHINIA to protect sensitive information like customer details and financial data.

In summary, the choice of an RDBMS for the BAUHINIA system will provide a robust, scalable, and efficient mechanism to store and manage the application's data. The use of SQL will ensure that the system can satisfy complex data needs, while built-in data integrity, consistency mechanisms, and security features will help maintain the reliability and security of the system.

1.6.4 Python and Django:

Python and Django represent a powerful combination of programming language and web framework, making them ideal for the development of the BAUHINIA system.

Python is a high-level, interpreted programming language known for its clear syntax and readability. It supports multiple programming paradigms, including object-oriented, imperative, and functional programming. Python's simplicity makes it a great choice for a project like BAUHINIA where rapid development is a priority. Here are some specific benefits of using Python for this project:

- Easy to Learn and Use: Python's syntax is designed to be readable and straightforward, which makes it easy to learn, read, and write.
- Robust Standard Library: Python's extensive standard library includes modules for everything from file I/O to system calls to Internet protocols. This "batteries included" philosophy means that often, complex tasks can be accomplished with just a few lines of Python code.
- Strong Support for Integration: Python can easily integrate with languages like C and C++, and it also provides powerful capabilities for controlling and performing complex operations on operating system services.

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Django takes care of much of the hassle of Web development, so we can focus on writing our app without needing to reinvent the wheel. Here's why Django is beneficial for BAUHINIA:

- Admin Interface: Django includes a built-in admin interface that makes it easy to build, inspect, and manage the data in our application.
- ORM Layer: Django's Object-Relational Mapping (ORM) layer is an excellent way to interact with our data. It's like a virtual database, providing a high-level, Pythonic interface to our data and abstracting the underlying database system.

- Security: Django provides good security measures by default, helping developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.
- Scalability: Django is designed to help developers take applications from concept to completion as quickly as possible. It provides a set of components to handle most web development tasks and can be scaled to handle high levels of traffic and data.

Overall, using Python and Django for the BAUHINIA system will allow for rapid development, ease of use, and a robust and secure final product. The combination of these tools will help to meet the system's functional requirements while also ensuring its non-functional requirements, like performance, security, and scalability, are well-managed.

1.6.5 Conclusion

In conclusion, the BAUHINIA system development process necessitates a comprehensive approach that takes into account business requirements, user requirements, system design, and the selection of appropriate software development tools and methodologies. By carrying out a thorough analysis of the problem at hand, it has been possible to devise a solution that not only meets the needs of the end-users but also aligns with the company's strategic goals.

- The Agile Scrum methodology was chosen over other approaches due to its flexibility, iterative nature, and focus on continuous improvement. This methodology allows for adaptive planning, encourages rapid and flexible response to changes, and provides a means to address issues or risks as they arise.
- Regarding system design, UML was chosen for its graphical representation of a system's architecture. Use Case, ERD, Class Diagrams, Sequence Diagrams, and Interface Designs are all integral parts of the system's design phase. The employment of UML allows the system's blueprints to be easily understood, reviewed, and modified, thus streamlining the development process.
- Python, with its readability and Django, its accompanying web framework, were selected for BAUHINIA's development due to the rapid development, clean design, and security they provide. This duo caters to BAUHINIA's requirements and is known for its efficiency, extensive library support, and ease of learning, all of which favor a speedy development and effective maintenance of the system.
- The use of a Relational Database Management System (RDBMS) is crucial due to the system's need to handle a significant volume of data and maintain data integrity. This choice is motivated by the robustness, scalability, and the data security features provided by RDBMS.

Each decision in the software design process was made with careful consideration, comparing between various tools and techniques. The selected methodologies, tools, and techniques are all aimed at ensuring that the BAUHINIA system is robust, efficient, and easy to use, offering a solution that optimally addresses the business problem at hand.

Activity 2

2.1 Software Development Tools

2.1.1 Introduction

The BAUHINIA software is a comprehensive inventory management application designed to cater to the multifaceted needs of modern businesses. The application offers robust functionalities for tracking inventory levels, orders, sales, and deliveries, with specific user requirements met for customers, inventory handling clerks, production managers, and chief accountants. Its implementation has significantly optimized the efficiency and effectiveness of inventory control within the organization, reducing errors and improving overall operational performance.

Developing such a complex and robust software system is a challenging task that requires careful planning, design, and implementation. The process involves understanding user requirements, designing the system architecture, coding, testing, deployment, and maintenance. Each stage of this development process requires the use of various software development tools and techniques to ensure that the final product meets the specified requirements and can be delivered on time and within budget.

Software development tools, often referred to as Computer-Aided Software Engineering (CASE) tools, play a crucial role in aiding developers to handle the complexity of software development. They provide automated support for the development process, making it easier to manage large codebases, track changes, test and debug the software, design user interfaces, manage project progress, and document the system. By enhancing productivity, improving quality, and reducing the time taken to develop software, these tools are a fundamental component of modern software development practices.

In the development of the BAUHINIA software, various CASE tools have been employed across the different stages of the software development lifecycle. These tools were instrumental in the successful implementation of the BAUHINIA software and its ability

to meet the specified user requirements. The following sections will delve deeper into the specific tools used and their contributions to the development of the BAUHINIA software.

2.2 Software Development Tools

2.2.1 Software Development Tools: Diagram Tools

Research on Various Diagram Tools:

Creating diagrams is a fundamental part of the software development process, as they help visualize complex systems, processes, and data structures. Various diagram tools are available to developers, each offering unique features and capabilities. Here are some that were considered for BAUHINIA's Application:

- Visio: A diagramming tool from Microsoft, used for creating flowcharts, network diagrams, org charts, floor plans, engineering designs, and more. It provides a wide array of diagramming features and integrates well with other Microsoft Office products.
- Lucidchart: A web-based diagramming tool that supports real-time collaboration. It's known for its user-friendly interface and variety of diagramming options.
- Draw.io (now diagrams.net): A free, online diagramming tool that's useful for creating simple diagrams. It offers straightforward features and integrates well with platforms like Google Drive and Confluence.
- UMLet: An open-source UML tool with a simple user interface. It's specifically designed for creating UML diagrams, making it ideal for designing software architectures.

Analysis and Comparison of Diagram Tools:

Each of the diagram tools researched offers unique features and capabilities:

- Visio is a robust tool with a comprehensive range of diagramming capabilities. Its integration with Microsoft Office is a major advantage for teams already using Microsoft products.
- Lucidchart shines with its collaborative features. It's ideal for teams looking for real-time collaboration and has a more intuitive interface compared to other tools.
- Draw.io is a good choice for teams on a budget, as it's free and provides enough functionality for most simple diagramming needs.
- UMLet, being specifically designed for UML diagrams, has features tailored towards software design. It's a good choice for complex software design tasks but may be overkill for simple diagramming needs.

Pros and Cons of Diagram Tools:

Here are some pros and cons for each tool:

- Visio
 - Pros: Wide range of features, integrates with Microsoft Office
 - Cons: Can be complex for beginners, costly for small teams
- Lucidchart
 - Pros: Real-time collaboration, user-friendly interface
 - Cons: Limited free version, subscription-based pricing can be costly
- Draw.io
 - Pros: Free, straightforward to use, integrates with various platforms
 - Cons: Limited features compared to other tools, less suitable for complex diagrams
- UMLet
 - Pros: Tailored for UML diagrams, free and open-source

- Cons: Limited to UML diagrams, interface isn't as intuitive

In summary, the best diagram tool depends on the specific needs and resources of the project. For BAUHINIA's Application, I found that a combination of tools like Visio for detailed design diagrams and Draw.io for simpler, collaborative diagrams worked best.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Microsoft Visio and Draw.io as the primary diagram tools.

Justification for Selected Tools:

1) Microsoft Visio:

Microsoft Visio was chosen because of its robust and versatile diagramming capabilities. As a part of the Microsoft ecosystem, it provides seamless integration with other Microsoft products like Word and PowerPoint, which are often used in software development projects. This allows for easy sharing and collaborative work within the team.

Visio's comprehensive suite of diagramming tools, including flowcharts, network diagrams, org charts, etc., offers the flexibility needed to design various aspects of the BAUHINIA application. From system architecture diagrams to detailed flowcharts of user interactions, Visio provides all the necessary tools.

2) Draw.io:

On the other hand, Draw.io was chosen for its simplicity and integration capabilities. The BAUHINIA project requires collaboration and real-time editing, both of which are supported by Draw.io. For less complex diagrams or initial conceptual sketches, Draw.io offers a quick and easy solution that can be accessed by anyone on the team.

Furthermore, as a free tool, Draw.io helps in keeping the project budget under control. It's a cost-effective solution for diagramming needs, especially for startups and small teams with limited resources.

In conclusion, the selection of Visio and Draw.io was based on the specific needs of the BAUHINIA project, considering factors like feature requirements, collaboration needs, integration with other tools, and budget constraints.

2.2.2 Software Development Tools: Project Management Tools

Research on Various Project Management Tools:

Project management tools are vital for planning, executing, and monitoring software development projects. They help teams collaborate, keep track of tasks, manage resources, and communicate effectively. Here are some project management tools that were considered for BAUHINIA's Inventory Control Application:

- Jira: A widely-used tool that supports agile development and provides features like issue and project tracking, customizable workflows, and reports.
- Trello: A Kanban-style project management tool that uses boards, lists, and cards to organize tasks. It's known for its simplicity and visual interface.
- Microsoft Project: A comprehensive project management software that offers features like scheduling, resource allocation, budget management, and reporting.
- Asana: A flexible tool that supports task management, team collaboration, and workflow management. It offers both list-style and Kanban-style views.

Analysis and Comparison of Project Management Tools:

Each of the project management tools researched offers unique features and capabilities:

- Jira is a versatile and powerful tool, particularly suited for agile teams. It's highly customizable and integrates well with other tools like Confluence and Bitbucket.
- Trello is known for its simplicity and visual interface. Its Kanban-style approach makes it suitable for managing simple or moderately complex projects.
- Microsoft Project offers comprehensive project management features, including advanced scheduling and resource management. It integrates well with other Microsoft products, but it may be too complex for small teams or simple projects.
- Asana provides a balance of simplicity and functionality, offering both list-style and Kanban-style views. Its flexibility makes it suitable for a wide range of projects.

Pros and Cons of Project Management Tools:

Here are some pros and cons for each tool:

- Jira
 - Pros: Robust features, customizable, good for agile teams
 - Cons: Can be complex for beginners, may be overkill for simple projects
- Trello
 - Pros: Easy to use, visual interface, good for simple projects
 - Cons: Limited advanced features, may not scale well for large projects
- Microsoft Project
 - Pros: Comprehensive features, good for complex projects
 - Cons: High learning curve, may be costly for small teams
- Asana
 - Pros: Flexible, easy to use, good for a wide range of projects
 - Cons: Advanced features require premium subscription

In summary, the best project management tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, I found that a combination of Jira for its robust tracking and agile capabilities, and Trello for its simplicity and visual task management, worked best.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Jira and Trello as the primary project management tools.

Justification for Selected Tools:

1) Jira:

Jira was chosen because of its robust and versatile project management capabilities. It is widely regarded as the go-to tool for agile software development due to its support for Scrum and Kanban methodologies. Jira's built-in issue and project tracking features allow teams to keep track of their progress in real time, which is crucial for managing a complex project like the BAUHINIA Inventory Control Application. Furthermore, Jira's ability to integrate with other tools such as Confluence for documentation and Bitbucket for code repository management provides a holistic project management solution.

2) Trello:

On the other hand, Trello was chosen for its simplicity and visual interface. Trello uses a board-based approach to project management, allowing team members to quickly and easily see the state of tasks and who is responsible for them. For simpler tasks or smaller teams within the BAUHINIA project, Trello offers a quick and easy-to-use solution that can be accessed by anyone on the team.

Using Trello in conjunction with Jira allows for more granular project management. For example, Trello can be used to manage individual tasks within a sprint that is being tracked on Jira. This offers the team flexibility in how they manage their workload while ensuring that they can keep track of the bigger picture.

In conclusion, the selection of Jira and Trello was based on the specific needs of the BAUHINIA project, considering factors like project complexity, team size, methodology, and collaboration needs. Together, these tools provide a comprehensive project management solution that caters to both the macro and micro needs of the team.

2.2.3 Software Development Tools: Programming Tools

Research on Various Programming Tools:

Programming tools are essential in any software development process. They offer features to assist developers in writing code, debugging, and optimizing performance. Here are some of the programming tools considered for BAUHINIA's Inventory Control Application:

- PyCharm: A powerful integrated development environment (IDE) for Python, offering features like intelligent code completion, on-the-fly error checking, quick-fixes, and automated code refactorings.
- Visual Studio Code: A lightweight, versatile code editor with support for Python and a wide array of other languages. It has an extensive marketplace for plugins, allowing developers to customize their environment.
- Jupyter Notebook: An open-source web application that allows creation and sharing of documents containing live code, equations, visualizations, and narrative text.
- Sublime Text: A sophisticated text editor for code, markup, and prose. It's known for its speed, ease of use, and powerful features.

Analysis and Comparison of Programming Tools:

Each of the programming tools researched offers unique features and capabilities:

- PyCharm: A full-featured IDE specifically designed for Python. It offers many tools for productive Python development, including a powerful debugger and test runner, Python profiler, a built-in terminal, integration with major VCS and built-in Database Tools.
- Visual Studio Code: A versatile code editor with robust extension support. It's highly customizable and can support many programming languages and frameworks with the right plugins.
- Jupyter Notebook: Perfect for prototyping and exploratory data analysis. It allows for interactive programming with quick feedback.

- Sublime Text: A fast and efficient text editor with a vibrant package ecosystem. It's very customizable and can handle very large files which can be beneficial in certain projects.

Pros and Cons of Programming Tools:

Here are some pros and cons for each tool:

- PyCharm
 - Pros: Powerful features for Python development, integration with major VCS, built-in terminal and database tools.
 - Cons: Resource-intensive, can be overkill for simple scripts or small projects.
- Visual Studio Code
 - Pros: Lightweight, versatile, extensive plugin marketplace.
 - Cons: Some advanced features require additional configuration.
- Jupyter Notebook
 - Pros: Interactive programming, useful for data analysis and prototyping.
 - Cons: Not suited for developing large applications or software products.
- Sublime Text
 - Pros: Fast and efficient, capable of handling large files, highly customizable.
 - Cons: Lack of certain features out of the box, some features require paid version.

In conclusion, the best programming tool for BAUHINIA's Inventory Control Application would depend on the specific needs and resources of the project. Given that I developed the application using Python and the Django framework, a combination of tools like PyCharm for in-depth development tasks and Visual Studio Code for lighter tasks might work best.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, PyCharm and Visual Studio Code were selected as the primary programming tools.

Justification for Selected Tools:

1) PyCharm:

PyCharm was selected for its powerful features specifically tailored for Python development. Since the BAUHINIA application was developed using Python and the Django framework, PyCharm's extensive capabilities in this area were invaluable.

PyCharm offers intelligent code completion, on-the-fly error checking, and automated code refactorings, which greatly enhance the efficiency and speed of development. Its powerful debugger and test runner, coupled with the Python profiler, are ideal for ensuring the robustness and performance of the application.

Moreover, PyCharm's built-in terminal and integration with major version control systems (VCS) streamline the development workflow. Its built-in database tools also come in handy when dealing with data-intensive applications like BAUHINIA.

2) Visual Studio Code:

On the other hand, Visual Studio Code was chosen for its lightweight design and versatility. It complements PyCharm by serving as a quick and easy tool for lighter tasks, such as simple scripting and text editing.

Visual Studio Code's extensive plugin marketplace allows it to support a wide array of programming languages and frameworks. This is advantageous in a project like BAUHINIA, where different aspects of the application may require different tools and technologies.

Furthermore, Visual Studio Code is an open-source tool, making it a cost-effective solution that aligns with the budget constraints of the BAUHINIA project.

In conclusion, the selection of PyCharm and Visual Studio Code was based on the specific needs of the BAUHINIA project, considering factors such as the application's technical requirements, the efficiency and speed of development, and budget considerations. These tools, when used in conjunction, provide a comprehensive and versatile development environment for the BAUHINIA application.

2.2.4 Software Development Tools: Web Development Tools

Research on Various Web Development Tools:

Web development is a critical aspect of any modern software application. There are various tools available to developers, each with its unique features and capabilities. Here are some that were considered for BAUHINIA's Inventory Control Application:

- Django: Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It's designed to help developers take applications from concept to completion as quickly as possible.
- React: React is a JavaScript library for building user interfaces, often used to build single-page applications or mobile applications. It's highly performant and allows for reusable components.
- Angular: Angular is a platform for building web applications. It combines declarative templates, dependency injection, end-to-end tooling, and integrated best practices to solve development challenges.
- Vue.js: Vue.js is a progressive JavaScript framework for building user interfaces. It is designed from the ground up to be incrementally adoptable, making it easy to integrate with existing projects or to use for building complex single-page applications.

Analysis and Comparison of Web Development Tools:

Each of the web development tools researched offers unique features and capabilities:

- Django: Django is a full-featured framework that includes almost everything a developer needs to build a web application. Its philosophy of "batteries included" means it comes with a lot of built-in features like an admin interface, ORM, authentication, and more.
- React: React is not a full framework but a library focused on the view layer. It's known for its efficient diffing algorithm and component-based architecture, making it performant and modular.

- Angular: Angular is a complete, full-featured framework like Django, but it is client-side and written in TypeScript. It's known for its powerful template system and command-line interface.
- Vue.js: Vue.js is also a progressive framework focused on the view layer like React, but it can also be gradually adopted to function as a full-featured framework like Angular.

Pros and Cons of Web Development Tools

Here are some pros and cons for each tool:

- Django
 - Pros: "Batteries included", mature and robust, great for rapidly building scalable web applications.
 - Cons: Not as performant for highly interactive UIs, might be overkill for simple applications.
- React
 - Pros: Highly performant, component-based, great for highly interactive UIs.
 - Cons: Only covers the view layer, often requires additional libraries for state management and routing.
- Angular
 - Pros: Full-featured, powerful template system, great for large-scale applications.
 - Cons: Steeper learning curve, can be verbose and complex.

- Vue.js
 - Pros: Easy to learn, incrementally adoptable, great for a variety of applications.
 - Cons: Smaller community, fewer resources compared to React and Angular.

In summary, the best web development tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, we found Django to be the best choice due to its robust built-in features and its excellent compatibility with Python, the primary programming language for this project.

Tool Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Django as the primary web development tool.

Justification for Selected Tool:

1) Django:

Django was chosen because it is a robust and versatile web development framework that facilitates rapid application development, which is a necessity for the BAUHINIA project. As a Python-based framework, Django integrates seamlessly with the Python programming language and libraries, the primary language used in the BAUHINIA project. This seamless integration simplifies the development process and helps to reduce potential compatibility issues.

The Django framework provides a comprehensive suite of web development tools, including an Object-Relational Mapper (ORM) for database access, middleware support for handling requests, and a built-in administrative interface, among other features. This "batteries-included" philosophy of Django means it comes with many built-in features that can accelerate the development process.

Another important factor for choosing Django is its strong security emphasis. It provides several built-in protections against common security threats such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and SQL Injection, which are critical for the security of BAUHINIA's Inventory Control Application.

In addition, Django's scalability and flexibility are significant for the BAUHINIA project as it allows for the future growth and evolution of the application.

In conclusion, Django was selected based on the specific needs of the BAUHINIA project, considering factors like rapid application development, seamless integration with Python, built-in functionalities, security, and scalability.

2.2.5 Software Development Tools: Configuration Management Tools

Research on Various Configuration Management Tools:

Configuration management is a vital aspect of software development. It allows developers to maintain consistency and track changes in the software's performance, functionality, and physical attributes. A variety of configuration management tools are available, each with unique features and capabilities. Some of the tools considered for BAUHINIA's Inventory Control Application are:

- Git: A distributed version control system, Git is widely used in software development for tracking changes in source code during software development. It is designed to handle everything from small to very large projects with speed and efficiency.
- Subversion (SVN): Another version control system that allows you to track changes to files and directories over time. Unlike Git, it's a centralized version control system, meaning all the version history is stored in a single place.
- Ansible: Ansible is a configuration management tool that provides a simple automation language that can perfectly describe IT application environments in Ansible Playbooks.
- Puppet: Puppet is another popular configuration management tool. It's designed to manage the configuration of Unix-like and Microsoft Windows systems declaratively.

Analysis and Comparison of Configuration Management Tools:

Each tool researched has unique features and capabilities:

- Git: Git is a robust tool used by millions of developers around the world. Its distributed architecture means that every developer has a full copy of the project history on their local machine, which allows for operation even when a central repository is not accessible.

- Subversion: Subversion(SVN) provides a simpler, more centralized model of version control, which can be more suitable for certain teams and projects. It also handles binary files more efficiently than Git.
- Ansible: Ansible shines for its simplicity and ease of use. Its playbooks are easy to write and understand, and it doesn't require any agents on the servers to perform its tasks.
- Puppet: Puppet is powerful and flexible, with a mature and extensive ecosystem. It uses a declarative language, allowing you to describe the desired state of your systems, while Puppet takes care of ensuring that state.

Pros and Cons of Configuration Management Tools:

Here are some pros and cons for each tool:

- Git
 - Pros: Distributed version control, fast and efficient, robust community support.
 - Cons: Complex to understand for beginners, handling of large binary files can be less efficient.
- Subversion (SVN)
 - Pros: Simple to understand, efficient handling of binary files.
 - Cons: Centralized version control can be a bottleneck, slower than Git.
- Ansible
 - Pros: Simple to use, agentless, YAML-based playbooks.
 - Cons: Less mature than other tools, some operations can be slower.
- Puppet

- Pros: Powerful and flexible, extensive community and ecosystem, mature product.
- Cons: Requires a dedicated server and agent software on the servers, higher learning curve.

In summary, the best configuration management tool depends on the specific needs and resources of the project.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Git and Ansible as the primary configuration management tools.

Justification for Selected Tools:

1) Git:

Git was chosen due to its robust capabilities and widespread adoption in the software development industry. The distributed nature of Git allows every developer on the team to have a complete local copy of the project history, thereby providing flexibility and independence.

Additionally, Git's powerful branching and merging capabilities allow for efficient management of different development versions, making it possible to work on multiple features or fixes simultaneously without affecting the main codebase. This feature is crucial for the BAUHINIA project, which will likely have various features being developed and bugs being fixed at any given time.

Lastly, Git offers excellent community support and integrates well with most of the popular development platforms, further solidifying its position as our choice of version control system.

2) Ansible:

Ansible was selected for its simplicity and ease of use. Its playbooks, written in YAML, are straightforward to understand and write, making it a more accessible tool for the team.

The agentless nature of Ansible is also a significant advantage. It doesn't require any specific software installed on the servers it manages, which makes it easier to maintain and reduces the overhead on the servers.

Furthermore, Ansible can manage complex multi-tier environments with ease, which is crucial for the BAUHINIA application that likely involves several interconnected components.

In conclusion, the selection of Git and Ansible was driven by their respective strengths and the specific needs of the BAUHINIA project. Factors such as ease of use, scalability, community support, and integration capabilities were all considered in this decision.

2.2.6 Software Development Tools: Quality Assurance Tools

Research on Various Quality Assurance Tools:

Quality Assurance (QA) is a critical component of software development, ensuring the delivery of robust, reliable, and high-quality software. Various QA tools are available to developers, each offering unique features and capabilities. Here are some that were considered for BAUHINIA's Inventory Control Application:

- JUnit: A testing framework for Java programming language, used for unit testing. It's simple, open-source, and widely adopted in the industry.
- Selenium: A popular tool for automated testing of web applications. It supports multiple languages and browsers, and allows for creating robust, browser-based regression automation suites and tests.
- Postman: A collaborative platform for API development and testing. It's used for testing APIs, documenting them, and simulating their behaviors before actual implementation.
- Jest: A JavaScript testing framework with a focus on simplicity. It supports different kinds of tests including snapshot testing, asynchronous testing, and manual mocking.

Analysis and Comparison of Quality Assurance Tools:

Each of the QA tools researched offers unique features and capabilities:

- Junit: JUnit is a widely used tool for unit testing in Java. Its simplicity and efficiency are key strengths.
- Selenium: Selenium shines in automated browser testing. Its ability to support multiple programming languages and browsers makes it versatile.
- Postman: Postman stands out for API testing. It simplifies the process of testing and documenting APIs, making it an ideal choice for web services and microservices.

- Jest: Jest, being tailored for JavaScript, is an excellent choice for testing JavaScript or TypeScript projects. It's simple to set up and use, and supports various types of testing.

Pros and Cons of Quality Assurance Tools:

Here are some pros and cons for each tool:

- JUnit
 - Pros: Simple, efficient, open-source
 - Cons: Limited to Java, doesn't support UI testing
- Selenium
 - Pros: Supports multiple languages and browsers, robust
 - Cons: Can be complex to set up, slow execution time
- Postman
 - Pros: Great for API testing, collaborative
 - Cons: Limited use outside of API testing, UI can be complex for beginners
- Jest
 - Pros: Easy setup, supports different types of testing
 - Cons: Limited to JavaScript, some report it's slower compared to other JavaScript testing frameworks

In summary, the best QA tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, the appropriate tools would depend on various factors such as the programming languages used, the types of testing required, and the team's familiarity with the tools.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I have selected JUnit and Selenium as the primary Quality Assurance tools.

Justification for Selected Tools:

1) JUnit:

JUnit was chosen for its simplicity, efficiency, and the widespread industry adoption. As BAUHINIA's application is built on Java, JUnit provides a reliable and efficient way to ensure the code behaves as expected under different conditions. Unit testing is a fundamental part of the software development process, as it allows developers to check that individual units of source code are working correctly. This promotes better design and allows for easier future maintenance, as the presence of unit tests can help developers identify and fix issues that may arise during the development process.

2) Selenium:

On the other hand, Selenium was chosen for its ability to automate browser activities, making it ideal for testing web applications like BAUHINIA's Inventory Control Application. Selenium provides a way to generate scripts for browser-based tests, which can simulate user interaction with the web application. This makes it possible to identify and fix UI/UX issues that may not be apparent in unit testing. The ability of Selenium to support multiple languages and browsers adds to its versatility and aligns with our need for comprehensive testing of the web interface across different platforms and browsers.

In conclusion, the selection of JUnit and Selenium was based on the specific needs of the BAUHINIA project, considering factors like the programming language used (Java), the nature of the application (web application), and the types of testing required (unit testing and browser-based testing). This combination of tools will ensure robust and comprehensive testing for the application, enhancing its reliability and performance.

2.2.7 Software Development Tools: Documentation Tools

Research on Various Documentation Tools:

Creating clear, thorough, and accessible documentation is a critical aspect of software development, facilitating understanding and collaboration within teams and ensuring effective use of the software by end-users. There are many documentation tools available, each offering unique features. Here are some that were considered for BAUHINIA's Inventory Control Application:

- Microsoft Word: A widely used text processing tool from Microsoft, often used for creating and sharing text documents. It's equipped with formatting tools, templates, and integrates well with other Microsoft Office products.
- Confluence: A collaboration tool from Atlassian, used to create, share, and collaborate on documents in real-time. It's often used for technical documentation in software projects.
- Markdown: A lightweight markup language used to create formatted text. It's popular in software development due to its simplicity and compatibility with version control systems like Git.
- Sphinx: A tool that makes it easy to create intelligent and beautiful documentation. It was originally created for Python documentation but can be used for other types of documentation as well.

Analysis and Comparison of Documentation Tools:

Each of the documentation tools researched offers unique features and capabilities:

- Microsoft Word: Microsoft Word is a robust tool with comprehensive text processing and formatting capabilities. Its integration with Microsoft Office is a major advantage for teams already using Microsoft products.
- Confluence: Confluence shines with its collaborative features. It's ideal for teams looking for real-time collaboration and offers document versioning, which is crucial in software development.

- Markdown: Markdown is a simple, straightforward option for teams looking to maintain documentation alongside code, as it can be version-controlled and viewed directly on platforms like GitHub.
- Sphinx: Sphinx is a powerful tool for creating detailed, structured technical documentation. It supports a range of output formats and integrates well with code, making it ideal for software projects.

Pros and Cons of Documentation Tools:

Here are some pros and cons for each tool:

- Microsoft Word
 - Pros: Wide range of features, integrates with Microsoft Office
 - Cons: Can be complex for beginners, not ideal for versioning or collaborative editing
- Confluence
 - Pros: Real-time collaboration, versioning, integrates with Jira and other Atlassian products
 - Cons: Subscription-based pricing can be costly, requires internet access for collaboration
- Markdown
 - Pros: Simple and straightforward, integrates well with version control systems
 - Cons: Limited formatting options, less suitable for complex documentation
- Sphinx
 - Pros: Supports structured, detailed documentation, integrates well with code

- Cons: Requires knowledge of reStructuredText, has a learning curve for beginners

In summary, the best documentation tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, a combination of tools like Microsoft Word for general documentation, Confluence for collaborative editing, and Markdown or Sphinx for technical documentation may be ideal.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Microsoft Word and Confluence as the primary documentation tools. Additionally, Markdown was chosen for maintaining documentation alongside the codebase.

Justification for Selected Tools:

1) Microsoft Word:

Microsoft Word was chosen due to its comprehensive text processing and formatting capabilities. As a part of the Microsoft ecosystem, it provides seamless integration with other Microsoft products, allowing for easy sharing and collaborative work within the team. Its wide range of features allows for the creation of various types of documents, from design documents to user manuals.

2) Confluence:

Confluence was chosen for its real-time collaboration and versioning features. It allows the team to work together on documents, maintaining a history of changes that provides traceability and accountability. Its integration with other Atlassian products, like Jira, also facilitates tracking the progress of tasks and linking documentation to relevant project tasks.

Markdown:

Markdown was chosen due to its simplicity and its ability to be version-controlled alongside the application's code. This is particularly useful for technical documentation that needs to stay closely aligned with the codebase. Markdown files can be viewed directly on platforms like GitHub, making it easily accessible to the development team.

In conclusion, the selection of Microsoft Word, Confluence, and Markdown was based on the specific needs of the BAUHINIA project, considering factors like the type of documentation needed, collaboration needs, integration with other tools, and the necessity of maintaining certain documentation alongside the codebase.

2.2.8 Software Development Tools: Design Tools

Research on Various Design Tools:

Designing interfaces and user experiences is a crucial part of the software development process. There are numerous tools available for this purpose, each with unique features and capabilities. Here are some that were considered for BAUHINIA's Inventory Control Application:

- Adobe XD: A user experience design tool for web applications and mobile apps. Adobe XD is part of Adobe's Creative Cloud and is used for wireframing, creating interactive prototypes, and testing.
- Figma: A web-based UI design and prototyping tool known for its real-time collaboration features. It allows multiple designers to work on the same design simultaneously.
- Sketch: A vector-based design tool focused on user interface and user experience design. It's widely used in the design community and has a large number of plugins and resources available.
- InVision: A prototyping tool used to create clickable and interactive prototypes. InVision also offers features for collaboration and feedback.

Analysis and Comparison of Design Tools:

Each of the design tools researched offers unique features and capabilities:

- Adobe XD is a robust tool with comprehensive capabilities for designing, prototyping, and testing UI/UX. It also integrates with other Adobe products, which can be an advantage for teams already using the Adobe suite.
- Figma is known for its real-time collaboration features, making it ideal for teams. It is web-based, which allows for platform-independent access, and its vector-based tools are capable of creating intricate interface designs.

- Sketch, while similar to Figma in terms of interface design capabilities, has a vast range of plugins that extend its functionalities. However, it's only available on MacOS.
- InVision allows you to create interactive and animated prototypes. It's less feature-rich for designing interfaces but integrates well with Sketch and Adobe XD for prototyping and collaboration.

Pros and Cons of Design Tools:

Here are some pros and cons for each tool:

- Adobe XD
 - Pros: Comprehensive design and prototyping tools, integrates with Adobe Suite.
 - Cons: Subscription-based, less collaborative than Figma.
- Figma
 - Pros: Real-time collaboration, web-based (platform-independent), strong community and resources.
 - Cons: Can be complex for beginners, free version has limitations.
- Sketch
 - Pros: Wide range of plugins, mature and robust.
 - Cons: MacOS only, less collaborative than Figma, requires paid license.
- InVision
 - Pros: Powerful prototyping and collaboration features, integrates with Sketch and XD.
 - Cons: Less comprehensive design tools, needs other design software for best results.

In summary, the best design tool depends on the specific needs and resources of the project. The BAUHINIA project would need to consider factors like collaboration requirements, platform limitations, design and prototyping features, and cost when choosing the right tool.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I have selected Figma and Adobe XD as the primary design tools.

Justification for Selected Tools:

1) Figma:

Figma was chosen because of its robust collaborative capabilities. Its real-time co-editing feature makes it ideal for the BAUHINIA project where team collaboration is a key requirement. As a web-based tool, Figma allows the team to access the design project from any device with internet access. This flexibility aids in fostering a productive and collaborative design process.

Figma also provides a strong suite of design and prototyping tools, making it an all-in-one solution for the design needs of the BAUHINIA project. Its vector-based tools are capable of creating intricate interface designs, and the prototyping features allow the team to create interactive previews of the application.

2) Adobe XD:

Adobe XD, on the other hand, was chosen for its comprehensive design and prototyping tools. Adobe XD offers a complete solution for designing, prototyping, and sharing interactive user interfaces. As part of the Adobe Creative Cloud, Adobe XD integrates seamlessly with other Adobe products, allowing for efficient workflows.

Furthermore, Adobe XD's auto-animate feature allows designers to create more complex and nuanced interactions. This level of detail can help the BAUHINIA team ensure the design meets the user's needs and expectations.

In conclusion, the selection of Figma and Adobe XD was based on the specific needs of the BAUHINIA project. The choice considered factors such as collaboration requirements, platform compatibility, the range of design and prototyping tools, and integration with other tools in the development stack.

2.2.9 Software Development Tools: Maintenance Tools

Research on Various Maintenance Tools:

Maintenance is an essential part of software development, ensuring that the application functions as expected and stays relevant to its users. Various tools exist to aid in this process, each with their own capabilities and features. For the BAUHINIA Inventory Control Application, I considered the following tools:

- Jira: A popular project management tool used widely in software development. Jira allows for tracking of issues, bugs, and tasks, making it a strong candidate for software maintenance.
- Bugzilla: An open-source tool specifically built for bug tracking and management. Bugzilla allows for detailed tracking and reporting of issues, which can be beneficial for maintenance.
- GitHub: While known as a version control platform, GitHub also has strong issue tracking and project management features, making it useful for maintenance tasks.
- Redmine: An open-source, web-based project management, and issue tracking tool. Redmine is flexible and customizable, providing a range of features useful for software maintenance.

Analysis and Comparison of Maintenance Tools:

Each of these maintenance tools offers unique features and capabilities:

- Jira: Jira is highly customizable and integrates with many other software development tools, making it a versatile choice for teams with complex needs.
- Bugzilla: Bugzilla is a specialist tool with a focus on bug tracking. It's less comprehensive than some others, but for teams needing a dedicated bug tracking tool, it's a strong choice.
- GitHub: GitHub is a familiar platform for many developers, and its issue tracking integrates seamlessly with the codebase. It's a convenient option for teams already using GitHub for version control.

- Redmine: Redmine provides a good balance of features, with both project management and issue tracking. As an open-source tool, it also offers a high degree of customizability.

Pros and Cons of Maintenance Tools:

Here are some pros and cons for each tool:

- Jira
 - Pros: Highly customizable, strong integrations
 - Cons: Can be complex to set up, costly for large teams
- Bugzilla
 - Pros: Focused on bug tracking, open-source
 - Cons: User interface is not as friendly, lacks broader project management features
- GitHub
 - Pros: Integrated with version control, familiar to many developers
 - Cons: Less feature-rich than dedicated project management tools
- Redmine
 - Pros: Balance of project management and issue tracking features, customizable
 - Cons: Setup and customization can be complex, user interface less polished

In summary, the best maintenance tool depends on the specific needs of the project. Factors such as the size of the team, the complexity of the project, and existing tool use can all influence the choice of maintenance tool for BAUHINIA's Inventory Control Application.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I have selected Jira and GitHub as the primary maintenance tools.

Justification for Selected Tools:

1) Jira:

Jira was chosen for its advanced and comprehensive project management capabilities. It is a feature-rich platform that provides detailed issue tracking, enabling teams to effectively manage and prioritize bugs and tasks. This will ensure smooth maintenance of the BAUHINIA application as the team can easily track ongoing issues and their resolutions.

In addition, Jira's robust integrations with other development tools like Bitbucket, Confluence, and Slack offer seamless connectivity and enhanced collaboration within the team. This makes Jira a highly versatile tool for managing not just maintenance tasks but also other aspects of software development.

2) GitHub:

GitHub, on the other hand, was chosen for its tight integration with the codebase and familiarity to most developers. The use of GitHub in BAUHINIA's development process ensures that any maintenance-related tasks, such as bug fixes or feature additions, can be directly linked to the relevant code changes. This tight coupling between the codebase and the maintenance tasks enhances traceability and accountability.

Moreover, the intuitive and developer-friendly nature of GitHub makes it an excellent tool for promoting team collaboration and efficiency. Developers can easily create and assign issues, link them to specific code changes, and discuss them directly on the platform.

In conclusion, the selection of Jira and GitHub was influenced by the specific needs of the BAUHINIA project, taking into consideration factors like advanced issue tracking, seamless integration with other tools, and enhancement of team collaboration. This

combination of tools offers a robust maintenance infrastructure for BAUHINIA's Inventory Control Application.

2.2.10 Software Development Tools: Analysis Tools

Research on Various Analysis Tools:

Analysis tools are critical for successful software development, helping developers understand and assess various aspects of the system. The following are some of the analysis tools considered for the BAUHINIA Inventory Control Application:

- Tableau: A powerful data visualization tool used for business intelligence. It can pull data from various data sources to create interactive dashboards, charts, and reports.
- RapidMiner: A data science platform that provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics.
- Google Analytics: A web analytics service that tracks and reports website traffic. It helps understand user behavior on the application, which is crucial for user experience improvement.
- Apache JMeter: An open-source software used for load testing and measuring performance. It helps identify performance bottlenecks in the system.

Analysis and Comparison of Analysis Tools:

Each of the analysis tools researched has unique features and capabilities:

- Tableau: Tableau is a comprehensive data visualization tool with robust features for creating interactive, real-time dashboards. Its wide range of data source connectivity makes it an excellent choice for diverse data environments.
- RapidMiner: RapidMiner provides a unified platform for various data analytics tasks. It's particularly useful if predictive analytics or machine learning is a significant part of the project.

- Google Analytics: Google Analytics is an invaluable tool for understanding user interaction with the application. It provides insights that are crucial for improving user experience and engagement.
- Apache JMeter: Apache JMeter shines in load testing and performance measurement. It's a key tool for ensuring the application's scalability and performance under high loads.

Pros and Cons of Analysis Tools:

Here are some pros and cons for each tool:

- Tableau
 - Pros: Powerful data visualization, integrates with many data sources.
 - Cons: Steep learning curve, can be costly for small teams.
- RapidMiner
 - Pros: Integrated platform for various data science tasks, offers a free version.
 - Cons: Can be complex for beginners, the full version is costly.
- Google Analytics
 - Pros: Provides valuable insights into user behavior, free to use.
 - Cons: Data privacy concerns, limited to web analytics.
- Apache JMeter
 - Pros: Excellent for load testing and performance measurement, free and open-source.
 - Cons: Requires knowledge of performance testing, limited to testing functions.

In summary, the best analysis tool depends on the specific needs and resources of the project. For the BAUHINIA Inventory Control Application, the combination of these tools would offer a broad spectrum of analytical capabilities, from data visualization to load testing.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Tableau, RapidMiner, and Google Analytics as the primary analysis tools.

Justification for Selected Tools:

1) Tableau:

Tableau was chosen for its powerful data visualization capabilities. Given that BAUHINIA's application handles a significant amount of inventory data, Tableau's ability to create interactive dashboards and reports provides invaluable insights into the inventory status and trends. Moreover, Tableau's capacity to pull data from various sources seamlessly ensures that the BAUHINIA team has access to the most updated data for decision making.

2) RapidMiner:

RapidMiner was chosen for its comprehensive data science platform. BAUHINIA's application requires not only data visualization but also predictive analytics to anticipate future inventory needs. RapidMiner's integrated environment for data preparation, machine learning, and predictive analytics meets these requirements, allowing the team to optimize inventory management based on predictive models.

3) Google Analytics:

Google Analytics was chosen to understand and enhance user interactions with the BAUHINIA application. As a web-based inventory application, understanding user behavior and improving user experience are crucial for BAUHINIA. Google Analytics

provides these insights, helping the team refine the application based on user behavior patterns.

In conclusion, the selection of Tableau, RapidMiner, and Google Analytics was based on the specific needs of the BAUHINIA project, considering factors such as data visualization, predictive analytics, user behavior analytics, and the integration of these tools with the existing system.

2.2.11 Software Development Tools: Process Modelling Tools

Research on Various Process Modelling Tools:

Process modelling is a crucial aspect of system analysis and design. It helps in visualizing the flow of data and control in the system. Various tools are available that cater to different aspects of process modelling. Here are some that were considered for BAUHINIA's Inventory Control Application:

- Bizagi: Bizagi is a popular business process modelling tool that offers capabilities like process design, documentation, simulation, and workflow automation. It uses BPMN (Business Process Model and Notation) for diagramming, making it an industry-standard tool.
- ARIS (Architecture of Integrated Information Systems): ARIS is a comprehensive tool for process modelling and enterprise architecture management. It offers a range of modelling methods and notations including EPC (Event-driven Process Chain), BPMN, and more.
- Visio: Microsoft's Visio, while primarily a diagramming tool, is also widely used for process modelling. It supports a variety of diagramming notations, including BPMN and flowcharts.
- Lucidchart: Similar to Visio, Lucidchart is a web-based diagramming tool that also supports process modelling with its extensive library of shapes and notations.

Analysis and Comparison of Process Modelling Tools:

Each of the tools researched offers unique features and capabilities:

- Bizagi: Bizagi is a robust tool designed specifically for process modelling and automation. Its simulation capability is a standout feature.
- ARIS: ARIS provides a comprehensive suite for enterprise architecture management in addition to process modelling, making it ideal for large, complex projects.

- Visio and Lucidchart: Visio and Lucidchart,, while not as specialized as Bizagi or ARIS, are versatile tools that can cater to various diagramming needs, including process modelling.

Pros and Cons of Process Modelling Tools:

- Bizagi
 - Pros: Comprehensive process modelling features, simulation capabilities
 - Cons: Can be complex for beginners, more suited to larger projects
- ARIS
 - Pros: Comprehensive enterprise architecture and process modelling tool, variety of modelling methods
 - Cons: High cost, steep learning curve
- Visio
 - Pros: Versatile, integrates with Microsoft Office
 - Cons: Limited specialized process modelling features, can be costly for small teams
- Lucidchart
 - Pros: User-friendly interface, real-time collaboration features
 - Cons: Limited free version, may lack advanced modelling capabilities

In summary, the best process modelling tool depends on the specific needs and resources of the project. Bizagi and ARIS provide comprehensive solutions for large, complex projects, while Visio and Lucidchart may be more suitable for smaller teams or simpler processes.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Bizagi and Microsoft Visio as the primary process modelling tools.

Justification for Selected Tools:

1) Bizagi:

Bizagi was chosen because of its comprehensive process modelling features and simulation capabilities. As a tool specifically designed for business process modelling, it offers advanced capabilities that make it an excellent choice for designing, documenting, and simulating processes in the BAUHINIA application. Furthermore, Bizagi's BPMN notation is an industry standard, ensuring that the process diagrams created with it can be easily understood by all stakeholders.

2) Microsoft Visio:

Visio, on the other hand, was selected for its versatility and integration with other Microsoft products. While it may not have the advanced process modelling capabilities of a tool like Bizagi, Visio provides enough features for general process modelling needs. The integration with other Microsoft products, such as Word and PowerPoint, makes it convenient for sharing and collaborative work.

In addition, Visio's support for various diagramming notations, including BPMN, ensures that it can cater to diverse diagramming requirements in the project.

In conclusion, the selection of Bizagi and Visio was based on the specific needs of the BAUHINIA project. Bizagi was chosen for its specialized process modelling capabilities and simulation features, while Visio was chosen for its general diagramming capabilities and integration with other tools. This combination of tools allows for comprehensive process modelling while keeping collaboration and integration needs in mind.

2.2.12 Software Development Tools: Prototyping Tools

Research on Various Prototyping Tools:

Prototyping tools are an essential part of the software development process, as they allow teams to visualize the user interface and functionality of an application before investing resources into development. Here are some that were considered for BAUHINIA's Inventory Control Application:

- Balsamiq: A rapid wireframing tool that helps to work faster and smarter. It reproduces the experience of sketching on a whiteboard but using a computer.
- Sketch: A design toolkit built to help you create your best work — from your earliest ideas, through to final artwork. It is primarily used for UI and UX design of mobile apps and web.
- Figma: A cloud-based design tool that is similar to Sketch in functionality and features, but with added advantages like cloud storage, real-time collaboration, and usability on multiple platforms.
- InVision: Offers design prototyping tool that allows for design sharing and collaboration. It has a range of features that help transform static designs into interactive and animated prototypes.
- Adobe XD: A vector-based user experience design tool for web and mobile applications. It's a part of Adobe's Creative Cloud Suite.

Analysis and Comparison of Prototyping Tools:

Each of the prototyping tools researched offers unique features and capabilities:

- Balsamiq: Balsamiq is known for its ease of use and rapid wireframing capabilities. It's a good tool for quickly creating design mockups and getting a feel for the user interface.
- Sketch: Sketch is a powerful design tool with a wide range of features and plugins. It is, however, limited to macOS.

- Figma: Figma shines with its real-time collaboration features. Its cloud-based nature means it's platform-independent and accessible from any web browser.
- InVision: InVision provides a range of capabilities from interactive prototyping to design sharing and collaboration, making it ideal for teams.
- Adobe XD: Adobe XD, as part of the Adobe Suite, integrates well with other Adobe products and offers a range of design and prototyping features.

Pros and Cons of Prototyping Tools:

Here are some pros and cons for each tool:

- Balsamiq
 - Pros: Easy to use, quick to create wireframes
 - Cons: Limited features for high-fidelity prototyping
- Sketch
 - Pros: Wide range of features, many plugins available
 - Cons: Limited to macOS, no real-time collaboration
- Figma
 - Pros: Real-time collaboration, platform-independent
 - Cons: Can be slower on low-end machines due to browser-based nature
- InVision
 - Pros: Good collaboration and sharing features, interactive prototyping
 - Cons: More complex to use compared to others
- Adobe XD
 - Pros: Integrates with Adobe Suite, wide range of features

- Cons: Learning curve can be steep for beginners

In summary, the best prototyping tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, the team will have to consider factors such as collaboration needs, operating system compatibility, and the level of fidelity required in the prototypes.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Figma and Balsamiq as the primary prototyping tools.

Justification for Selected Tools:

1) Figma:

Figma was chosen due to its robust functionality and collaborative capabilities. As a cloud-based platform, it enables real-time collaboration, making it easy for team members to work together regardless of their location. Figma's versatile features allow for both low-fidelity and high-fidelity prototyping, accommodating a broad range of design needs for BAUHINIA's Inventory Control Application.

Figma's platform independence is another significant advantage, as it allows design access and modification from any device with an internet connection. This feature facilitates a flexible work environment and ensures design continuity even if team members use different operating systems.

2) Balsamiq:

On the other hand, Balsamiq was chosen for its simplicity and speed. The low-fidelity wireframes Balsamiq creates are perfect for the initial stages of the design process when the team is still brainstorming and iterating on basic design concepts.

Balsamiq's user-friendly interface and easy-to-use features allow for quick sketching of UI elements, which is ideal for generating and discussing ideas within the team. The ease and speed at which Balsamiq can produce wireframes help to facilitate discussions, convey ideas, and garner feedback early in the design process.

In conclusion, the selection of Figma and Balsamiq was based on the specific needs of the BAUHINIA project, taking into consideration factors such as collaboration requirements, fidelity of prototypes, flexibility of access, speed of prototyping, and budget constraints.

2.2.13 Software Development Tools: Change Control Tools

Research on Various Change Control Tools:

Change control tools play a crucial role in managing the changes during software development. They help ensure that all changes are properly reviewed, approved, and implemented without adversely affecting the existing system. Here are some change control tools that were considered for BAUHINIA's Inventory Control Application:

- Jira: A popular tool from Atlassian, Jira is widely used for issue tracking, project management, and change control. It's flexible and customizable, allowing teams to adapt its use to their specific workflows.
- Git: Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It allows multiple developers to work on the same codebase without overwriting each other's changes.
- IBM Rational ClearQuest: This is a comprehensive software change management system that provides change tracking, process automation, reporting, and lifecycle traceability. ClearQuest offers robust features for enterprise-scale projects.
- ServiceNow: ServiceNow offers a change management module that provides a system of record for IT infrastructure, enabling IT departments to control IT service changes effectively.

Analysis and Comparison of Change Control Tools:

Each of the change control tools researched offers unique features and capabilities:

- Jira: Jira is well-regarded for its customization capabilities and its integration with other Atlassian tools, such as Confluence and Bitbucket.
- Git: Git, being a version control system, shines in its ability to track and manage changes to code. It's essential for teams that have multiple developers working on the same project.

- IBM Rational ClearQuest: IBM Rational ClearQuest provides comprehensive features that are beneficial for larger, enterprise-scale projects. Its automation capabilities are also a significant advantage.
- ServiceNow: ServiceNow is known for its ability to provide a consolidated view of all IT services, which can be beneficial for IT change management.

Pros and Cons of Change Control Tools:

Here are some pros and cons for each tool:

- Jira
 - Pros: Highly customizable, integrates with other Atlassian tools
 - Cons: Can be complex to set up and use, subscription-based pricing
- Git
 - Pros: Free and open-source, excellent for code version control
 - Cons: Can have a steep learning curve, primarily a code management tool
- IBM Rational ClearQuest
 - Pros: Comprehensive and robust features, excellent for large projects
 - Cons: Can be complex to use, costly for small teams
- ServiceNow
 - Pros: Consolidated view of all IT services, good for IT change management
 - Cons: Subscription-based pricing, might be overkill for smaller projects

In summary, the best change control tool depends on the specific needs and resources of the project. For BAUHINIA's Inventory Control Application, factors such as the scale of the project, the team's familiarity with the tools, and the budget would need to be considered.

Tools Selected for BAUHINIA's Application:

For the BAUHINIA Inventory Control Application, I selected Jira and Git as the primary change control tools.

Justification for Selected Tools:

1) Jira:

Jira was chosen because of its robust change management capabilities and customization options. As part of the Atlassian ecosystem, it integrates seamlessly with other tools used in software development, such as Confluence and Bitbucket, which provides a consolidated view of the project.

The custom workflows in Jira make it highly adaptable to different project management methodologies, like Scrum, Kanban, or even a mix. Also, the issue tracking feature of Jira can be used to keep track of change requests, bugs, and other issues, which could be crucial for managing changes in BAUHINIA's Inventory Control Application.

2) Git:

On the other hand, Git was selected for its unparalleled version control capabilities. With multiple developers working on the BAUHINIA Inventory Control Application, it's important to have a robust system to manage changes to the codebase.

Git's features like branching and merging allow developers to work on different features or bug fixes simultaneously without disrupting the main code. This enables a smooth workflow and efficient collaboration among the development team.

In conclusion, the selection of Jira and Git was based on the specific needs of the BAUHINIA project. Both tools are scalable, facilitating management as the project grows. They were selected keeping in mind factors like feature requirements, collaboration needs, and the scale of the project.

2.2.14 Comparison differences between each Software Development Tools

Table 2. 1 Comparison differences between each Software Development Tools

Software Development Tools	Key Features	Strengths	Weaknesses
Diagram Tools	Graphical representation of data.	Facilitates understanding, good for planning.	May require learning curve, can be time-consuming.
Project Management Tools	Manage tasks, resources, timelines.	Organizes work, facilitates collaboration.	May be complex to set up, requires commitment.
Programming Tools	Includes IDEs, libraries, compilers.	Speeds up development, automates repetitive tasks.	Can be complex, some tools may be costly.
Web Development Tools	Used for building websites and web apps.	Allows quick prototyping, supports various technologies.	Can have steep learning curve, some tools may have compatibility issues.
Configuration Management Tools	Track & control changes in the software.	Enhances collaboration, reduces conflicts.	Can be complex to set up, requires discipline.
Quality Assurance Tools	Facilitate testing and bug tracking.	Improves software quality, helps prevent issues.	May require trained personnel, some tools can be expensive.
Documentation Tools	Create and manage software documentation.	Improves communication, essential for maintenance.	Can be time-consuming, often overlooked.
Design Tools	Assist in creating software designs	Promotes better design, helps visualize concepts	May require learning curve, can be costly

Maintenance Tools	Assist in updating, improving, and fixing software.	Ensures software longevity, improves user satisfaction.	May be overlooked, requires ongoing effort.
Analysis Tools	Analyze code, performance, or user behavior.	Helps identify issues, optimize performance.	Can be complex, may require skilled personnel.
Process Modelling Tools	Visualize and analyze processes.	Helps optimize processes, good for planning.	May require learning curve, may not reflect reality.
Prototyping Tools	Create simplified versions of software.	Allows for early feedback, reduces risk.	Can be time-consuming, may lead to scope creep.
Change Control Tools	Manage changes in code and documents.	Ensures stability, improves collaboration.	Can be complex to set up, requires discipline.

2.3 Software Development Techniques

2.3.1 Types of software development Techniques

1. Traditional or Sequential Models:

These are the models where one phase of the project must be completed before moving onto the next, with little to no overlap between the phases.

- Waterfall Model

2. Iterative and Incremental Development:

These methodologies involve breaking down the project into smaller pieces, which are then developed in cycles, each improving upon the last.

- Spiral Model
- Rapid Application Development (RAD)

3. Agile Methods:

Agile methodologies emphasize flexibility, collaboration, customer involvement, and adaptability throughout the development process.

- Agile
- Scrum
- Extreme Programming (XP)
- Lean Software Development
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)

4. Prototype Based Models:

These models focus on creating a simplified version of the software, known as a prototype, before building the final product.

- Prototype Methodology

5. Collaborative Development Models:

These methodologies emphasize the collaborative efforts of cross-functional teams and involve stakeholders in the process.

- Joint Application Development (JAD)

6. Test-Driven Development:

These methodologies emphasize the writing of tests before the development of the actual software.

- Test-Driven Development (TDD)
- Behavior Driven Development (BDD)

It's important to note that some methodologies may fall under multiple categories based on their principles and practices. For example, Scrum and Extreme Programming (XP) are subsets of Agile, but they also incorporate iterative development and collaborative approaches.

2.3.2 Software Development Techniques: Waterfall Model

What is the Waterfall Model?

The Waterfall Model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion, like a waterfall. The phases in the waterfall model are Requirement Gathering and Analysis, System Design, Implementation, System Testing, Deployment, and Maintenance.

What makes the Waterfall Model special compared to other techniques?

The Waterfall Model is one of the simplest, most straightforward, and oldest models of software development. Its sequential nature makes it easy to understand and use. Unlike many other methodologies, each phase in the Waterfall Model is completed fully before the next one is started. This model is more disciplined because it has a clear structure with distinct goals for each phase.

Pros of the Waterfall Model

- Easy to Understand and Use: The Waterfall Model's straightforward, linear nature makes it easy to understand and use, even for those new to software development.
- Defined Stages and Deliverables: Each phase has specific deliverables and a review process. This allows for clear understanding of what is expected and when.
- Document-Driven: It requires heavy documentation upfront, ensuring that everything is well-documented which can be useful for future reference and for new team members to understand the project.
- Good for Simple, Clear Projects: It works well for projects where requirements are well-understood and unlikely to change.

Cons of the Waterfall Model

- Difficult to Change: Once a stage is completed, it's hard and costly to go back and change something. This is a major disadvantage if the project is long and complex.
- Assumes Requirements are Well-Understood Early: It's not suitable for complex and object-oriented projects or where requirements are at a moderate to high risk of changing.
- Testing is Delayed: As testing is done after the completion of all coding processes, bugs may be discovered late in the process when they can be expensive to fix.
- Long Delivery Times: Since the product is only delivered after all phases are complete, it may not be suitable for projects with tight timelines.
- Lack of Flexibility: The model is not suitable for the projects where requirements are not clear or not well understood at the beginning.

Overall, the Waterfall Model is an excellent choice for small, well-defined projects. However, for larger or more complex projects, or those with changing requirements, other methodologies like Agile or Spiral may be more appropriate.

2.3.3 Software Development Techniques: Prototype Methodology

What is the Prototype Methodology?

The Prototype Methodology is a software development process which allows developers to create only the prototype of the solution to demonstrate the functionality to the clients and make necessary modifications before developing the actual application. It begins with requirements gathering, where developers interact with the clients to understand their expectations and requirements. Developers then create a rough prototype of the solution, which is reviewed by the client and used to refine requirements. Once the client is satisfied with the prototype, developers use it to build the final system.

Why is the Prototype Methodology special compared to other techniques?

Unlike the Waterfall Model, where the whole software product is delivered after a long development period, the Prototype Methodology enables early feedback from the user because a working model of the system is provided early in the process. This ensures that the end product is more likely to meet the user's expectations and needs. This methodology allows developers to detect and fix any design flaws early in the development process, which can save time and money later.

Pros of the Prototype Methodology

- Early Feedback and Detection of Misunderstandings: The client and the development team can detect issues early in the development cycle. This can lead to improvements and changes in the requirements which reduce unnecessary efforts and rework later.
- Improved User Involvement and Satisfaction: The methodology allows users to see, and interact with, a working model of the software early in the project, which can help improve user satisfaction.

- Better Requirements Understanding: Developers get a clearer understanding of what needs to be done and what the system should look like, leading to better end results.

Cons of the Prototype Methodology

- Insufficient Analysis: Developers may undertake insufficient analysis of the project in their hurry to get a prototype to the client. This can lead to overlooking better or more efficient solutions.
- User Confusion: Users may think that a complete system is developed and start demanding more than was originally planned. This might lead to scope creep, causing the project to go beyond its initial timeline or budget.
- Over-Reliance on Prototypes: Some teams might focus too much on developing a prototype and compromise on the quality of the final product.
- Initial Cost: Developing a prototype can be time-consuming and costly initially, especially if the requirements are complex.

In summary, Prototype Methodology is excellent for projects where requirements are unclear and likely to change, and it is important to involve users in the development process early. However, it may not be suitable for projects with well-understood requirements and constraints, or where building a prototype might be overly time-consuming or costly.

2.3.4 Software Development Techniques: Spiral Model

What is the Spiral Model?

The Spiral Model is a risk-driven software development process model that is a combination of iterative development and the systematic, controlled aspects of the Waterfall Model. This model allows for multiple rounds (or spirals) of refinement, with each spiral consisting of four phases:

1. Identify: Determine objectives, alternatives, and constraints of the iteration.
2. Design: Evaluate alternatives, identify and resolve risks, and develop deliverables.
3. Build: Develop and verify the deliverable product.
4. Evaluate: Plan the next iteration, commit to an approach for it, and review the results of the current iteration.

The process continues in subsequent spirals until the final product is ready.

Why is the Spiral Model special compared to other techniques?

The Spiral Model is specifically designed to minimize risk. At the heart of this model is a focus on incremental releases and iterations, which means that a portion of the system is developed and delivered to the client to ensure that development is on the right track. It differs from other methodologies like the Waterfall and Prototype methodologies, which usually require more significant portions of the project to be completed before they can be reviewed or tested.

Pros of the Spiral Model

- Risk Analysis: Emphasizing risk analysis, this model helps teams avoid significant challenges or project failures.

- Flexibility: The Spiral Model is adaptable and flexible to changes. It allows for iterations and can adjust for more accurate requirements and specifications as they are understood.
- Client Engagement: The model involves the client in the development process, enhancing communication and ensuring requirements are met.

Cons of the Spiral Model

- Requires Risk Assessment Expertise: The team must be proficient in identifying and evaluating risks, which might not always be possible.
- Complex: This model is more complex than other models like Waterfall or Agile, which might make it harder to understand and manage.
- Expensive: The model's emphasis on risk analysis and customer interaction can make it a costly option compared to other methodologies.

In summary, the Spiral Model is well-suited for large, complex, and high-risk projects. However, for smaller projects with well-defined requirements, simpler methodologies like Agile or Waterfall might be more suitable and cost-effective.

2.3.5 Software Development Techniques: Agile Methodology

What is Agile Methodology?

Agile Methodology is an iterative approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. It's characterized by short development cycles called "sprints," typically lasting two weeks to a month, at the end of which a usable increment of the software is delivered.

In Agile development, requirements and solutions evolve through collaborative effort among cross-functional teams. Agile promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages flexible responses to change.

Why is Agile special when compared to other techniques?

Unlike traditional methodologies like Waterfall or Spiral, Agile is highly adaptable and allows changes to be made after the initial planning stage. It promotes close collaboration with customers and continuous input from end-users. Agile encourages iterative progress, frequent feedback, and quick adaptation to changes, making it uniquely suited for projects where requirements are expected to change or are not fully understood at the project's outset.

Pros of Agile Methodology

- **Flexibility and Adaptability:** Agile is highly adaptable to changes. As requirements evolve, teams can adjust their efforts to deliver the most valuable features first.
- **Customer Satisfaction:** Agile often leads to higher customer satisfaction as it allows for regular communication, feedback, and adjustments to meet the customer's needs effectively.
- **Risk Management:** Regular iterations and frequent testing mean that issues can be identified and addressed quickly, reducing the risk of project failure.

Cons of Agile Methodology

- Requires Experience: To work effectively, the Agile methodology requires a team experienced in Agile practices. Without proper experience, the project can quickly become unmanageable.
- Less Predictability: Agile projects can be less predictable in terms of timelines and final product, especially if the scope is not well defined or controlled.
- Not Suitable for All Types of Projects: Some projects, particularly those with a well-defined, unchanging set of requirements, may not benefit from the Agile approach.

In summary, Agile is a flexible and customer-centered approach to software development that works well for projects where requirements are expected to change. However, it requires an experienced team and a certain level of customer engagement, which might not be feasible or necessary for all types of projects.

2.3.6 Software Development Techniques: Scrum

What is Scrum?

Scrum is an Agile development framework that is iterative and incremental. It emphasizes collaboration, team self-management, and the flexibility to adapt to emerging business realities. The three key roles in a Scrum team are the Product Owner (who defines the product in detail), the Scrum Master (who ensures everyone follows Scrum principles and practices), and the Development Team (who design and build the software).

The development process is divided into "Sprints", typically lasting two weeks to one month, during which a usable increment of the software is built. The progress is tracked with daily meetings (Daily Scrum) and at the end of each Sprint, the team conducts a Sprint Review and a Sprint Retrospective to analyze what was done and how the process can be improved for the next Sprint.

Why is Scrum special when compared to other techniques?

Unlike traditional project management approaches, Scrum encourages active user involvement, multifunctional teams, and a collaborative approach to tasks. It also allows for changes to be made during the development process based on iterative feedback, leading to a better final product that aligns more closely with customer needs. Compared to other Agile techniques, Scrum provides a detailed set of roles and practices to guide the team.

Pros of Scrum

- **High Product Quality:** Regular checks and iterative feedback during Scrum development mean that errors are identified and corrected more quickly.
- **Enhanced Customer Satisfaction:** The customer is involved throughout the development process, which increases the likelihood that the final product will meet their expectations.
- **Increased Team Morale:** Scrum often results in higher team morale due to its emphasis on input and collaboration from all team members.

Cons of Scrum

- Not Suitable for Certain Types of Projects: Scrum can be less effective for projects with a clear, unchanging set of requirements or for projects where the team is not co-located.
- Requires Experienced Commitment: Scrum relies heavily on the commitment and cooperation of the whole team. If members are not fully committed or experienced, the project may not be successful.
- Potential for Scope Creep: Without a careful management, the scope of the project can expand beyond the original goals due to the flexibility of the methodology.

In summary, Scrum is a flexible and collaborative approach that works well for software development teams that can commit to the structure and responsibilities of the Scrum roles. However, like all methodologies, its effectiveness depends on the nature of the project and the team's experience and commitment.

2.3.7 Software Development Techniques: Extreme Programming (XP)

What is Extreme Programming (XP)?

Extreme Programming (XP) is an Agile software development methodology aimed at improving software quality and responsiveness to changing customer requirements. As a type of Agile method, XP advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

XP consists of various practices such as pair programming, extensive code review, unit testing of all code, simplicity and clarity in code, expecting changes in the customer's requirements and the use of metaphor to guide development.

Why is XP special when compared to other techniques?

XP stands out from other methodologies through its early, concrete, and continuing feedback from short cycles, its focus on teamwork, and the flexible use of various practices to fit a particular project. Its commitment to maintaining high-quality, simple code from the very beginning of a project also distinguishes XP. It's an approach that stresses communication and collaboration, and values working software over comprehensive documentation.

Pros of Extreme Programming (XP)

- Improved Software Quality: XP's emphasis on testing and review leads to fewer bugs and higher-quality software.
- Customer Satisfaction: By encouraging customer involvement and accommodating changes, XP ensures that the product more closely meets the user's needs.
- Teamwork and Collaboration: XP practices such as pair programming and collective code ownership encourage teamwork and result in a more cohesive and balanced team effort.

Cons of Extreme Programming (XP)

- Dependence on User Involvement: XP heavily relies on continuous user involvement, which may not always be practical or feasible.
- Excessive Overhead: Practices like pair programming could lead to higher costs, and the continuous involvement and communication required may be seen as overhead.
- Less Documentation: Due to its focus on coding, XP may neglect documentation, which can be problematic in terms of future maintenance or handovers.

In summary, XP offers a set of practices that can lead to high-quality software and a high degree of customer satisfaction. However, it requires a firm commitment to its practices and values, and it may not be suitable for all types of projects or teams.

2.3.8 Software Development Techniques: Lean Software Development

What is Lean Software Development?

Lean Software Development (LSD) is an iterative Agile methodology that focuses on the efficiency of resources and elimination of waste to deliver software products quickly. Originating from lean manufacturing principles (from Toyota's production system), it was adapted to software development by Mary and Tom Poppendieck.

Lean principles include:

1. Eliminate waste: Remove anything not adding value to the customer.
2. Amplify learning: Increase knowledge and skills to deliver better, faster solutions.
3. Decide as late as possible: Keep options open for as long as practical to allow for flexibility.
4. Deliver as fast as possible: Quicker delivery accelerates customer feedback and potential returns.
5. Empower the team: Value the team's ability to make decisions and solve problems.
6. Build integrity in: Create a consistent and cohesive system.
7. See the whole: Understand the larger context and system in which you are operating.

Why is LSD Special When Compared to Other Techniques?

LSD is special because it emphasizes optimizing efficiency by reducing waste. Unlike other techniques that focus mainly on managing project activities, LSD looks at the process holistically and finds ways to streamline and improve it continuously. This approach leads to quick, efficient delivery of high-quality software.

Pros of Lean Software Development

- Efficiency and Speed: By focusing on eliminating waste and streamlining processes, LSD can deliver products faster and with less effort.
- Flexibility: The principle of "decide as late as possible" allows for more adaptability in the face of changing requirements.
- Empowered Teams: LSD encourages the decentralization of decision-making, leading to more engaged, empowered teams.

Cons of Lean Software Development

- Requires a Shift in Mindset: For many organizations, implementing lean principles requires a significant cultural change, which can be difficult.
- Risk of Overlooking Details: The focus on speed and efficiency might cause some teams to overlook certain details, potentially impacting product quality.
- Dependence on the Entire Team: Lean is a team-oriented process where everyone's participation matters. If a member is not engaged or fails to perform, it can impact the whole project.

Overall, Lean Software Development can provide significant benefits in terms of efficiency and speed, but its success heavily depends on the team's mindset and commitment to lean principles.

2.3.9 Software Development Techniques: Rapid Application Development (RAD)

What is Rapid Application Development (RAD)?

Rapid Application Development (RAD) is a type of agile software development methodology that prioritizes rapid prototype releases and iterations. It aims to improve speed and flexibility in the development process by using a combination of structured techniques and prototyping to define users' requirements and design the final system.

The RAD model has four stages:

1. Requirements planning: Defining the business needs, project scope, constraints, and system requirements.
2. User design: User involvement in the system design through iterative prototyping and feedback.
3. Rapid construction: Coding and application development, and continuous integration.
4. Cutover: Final system testing, user training, and system deployment.

Why is RAD Special When Compared to Other Techniques?

RAD is special because of its emphasis on speed and user involvement. Unlike traditional methods, such as the Waterfall model, which focus on extensive upfront planning and sequential development stages, RAD aims to deliver functional prototypes as quickly as possible, and iteratively refines them based on user feedback until the final system meets the business needs. This approach can lead to a higher level of customer satisfaction and improved system usability.

Pros of Rapid Application Development (RAD)

- Speed: RAD allows for faster development and delivery by using prototyping and iterative development.
- User involvement: Users are involved throughout the process, helping to ensure that the final product meets their needs and expectations.
- Flexibility: The iterative nature of RAD allows for changes to be made more easily than in traditional development methodologies.

Cons of Rapid Application Development (RAD)

- Requires skilled developers: The fast-paced nature of RAD requires a team of experienced developers who can create high-quality systems quickly.
- Not suitable for all projects: RAD is ideal for projects with clear objectives and stable requirements. It may not work as well for larger, more complex projects or those with volatile requirements.
- Limited documentation: In the rush to create prototypes and complete the final product, documentation may be neglected, which can lead to issues with future maintenance or upgrades.

Overall, Rapid Application Development can offer significant benefits in terms of speed and customer satisfaction, but it requires the right project conditions and a skilled team to be effective.

2.3.10 Software Development Techniques: Joint Application Development (JAD)

What is Joint Application Development (JAD)?

Joint Application Development (JAD) is a process used in the systems development life cycle to gather business requirements while developing new information systems for a company. The JAD approach involves continuous interaction with the users and different stakeholders of the system to gather requirements in a collaborative and interactive manner.

JAD primarily focuses on a series of collaborative workshops (JAD sessions) involving various system stakeholders, such as product owners, users, designers, analysts, and developers. These workshops aim to result in faster, more accurate results due to the synergistic problem-solving approach.

Why is JAD Special When Compared to Other Techniques?

Compared to other methods, JAD is more collaborative and interactive. It emphasizes direct communication among stakeholders and continuous user involvement. This leads to higher user acceptance, as users and other stakeholders have a say in the development from the very beginning.

JAD sessions help in understanding the system from different perspectives, leading to more comprehensive and accurate requirements. This, in turn, results in better design decisions and a higher quality end product.

Pros of Joint Application Development (JAD)

- Increased User Involvement and Ownership: JAD encourages the active involvement of users from the beginning, which leads to a higher sense of ownership and acceptance of the final product.
- Improved Requirements Understanding: Because of the collaborative workshops, stakeholders can better understand the system requirements from different perspectives.

- Reduced Development Time: By capturing accurate requirements up front, the system development time can be reduced as it minimizes the amount of rework.

Cons of Joint Application Development (JAD)

- Requires Commitment: For JAD to be successful, it requires a high level of commitment from all participants, and it can be time-consuming.
- Not Suitable for All Projects: JAD might not be suitable for projects where it's challenging to bring all key stakeholders together for the sessions due to geographical, time, or resource constraints.
- Dependent on Facilitator: The success of JAD sessions heavily relies on the skills of the facilitator. A poorly facilitated session might lead to misunderstanding, conflicts, or ineffective results.

In summary, Joint Application Development can be an effective technique to gather and consolidate requirements in a collaborative manner. However, it requires careful planning, skilled facilitation, and commitment from all stakeholders.

2.3.11 Software Development Techniques: Dynamic Systems Development Method (DSDM)

What is the Dynamic Systems Development Method (DSDM)?

The Dynamic Systems Development Method (DSDM) is an Agile project delivery framework, primarily used as a software development method. First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method. Over time, it has matured and evolved into a comprehensive framework for Agile project management and delivery.

DSDM is based on nine key principles that primarily revolve around business needs/value, active user involvement, empowered teams, frequent delivery, integrated testing, and stakeholder collaboration. It prescribes a detailed process with specific phases (Pre-project, Feasibility, Foundations, Exploration, Engineering, Deployment) and roles for team members.

Why is DSDM Special When Compared to Other Techniques?

Unlike some other Agile methods, DSDM integrates project management and product development into a single process, providing more structure and formalization than methods like Scrum or XP.

Another differentiating factor is DSDM's strong focus on aligning projects with business strategy and delivering real business benefits, even if that means changing the scope of the project, as opposed to delivering all features as initially planned.

Pros of Dynamic Systems Development Method (DSDM)

- **Strong Business Focus:** DSDM has a robust focus on aligning the software product with business needs and ensuring that every iteration delivers real business benefits.
- **Built-In Project Management:** The DSDM framework includes principles and practices for project management, which is not a feature of all Agile methods.

- Frequent and Incremental Delivery: DSDM emphasizes delivering working software quickly and improving it incrementally. This allows for early ROI and continuous improvement.

Cons of Dynamic Systems Development Method (DSDM)

- Requires Training and Skill: Implementing DSDM requires a significant understanding of the framework, and team members might require training. Furthermore, DSDM's success largely depends on the skill and competence of the team members.
- Limited Flexibility: Although DSDM is an Agile method, it's more prescriptive and less flexible than other Agile frameworks like Scrum.
- Requires Full Stakeholder Commitment: DSDM's iterative approach requires constant collaboration and feedback from all stakeholders. If any key stakeholder is not fully committed, the project may face significant hurdles.

In conclusion, DSDM can be an effective software development technique for organizations looking for a more structured Agile method that integrates project management and product development and emphasizes business value. However, it requires a well-trained team and strong commitment from all stakeholders.

2.3.12 Software Development Techniques: Feature Driven Development (FDD)

What is Feature Driven Development (FDD)?

Feature Driven Development (FDD) is an iterative and incremental software development process that follows the principles of the Agile methodology. It was originally designed by Jeff De Luca in the late 1990s.

FDD emphasizes the delivery of tangible, working software in a timely manner. It breaks down the project into individual features, small enough to be implemented in a few weeks. The development of these features is then driven by feature lists which are constantly updated and prioritized. FDD has five key activities: Develop an Overall Model, Build a Features List, Plan by Feature, Design by Feature, and Build by Feature.

Why is FDD Special When Compared to Other Techniques?

FDD is particularly efficient in managing large-scale projects with multiple teams. It's highly focused on specific features, allowing for a more precise tracking of progress and productivity. It is also unique in the sense that it combines the quick iterative style of Agile with precise tracking and documentation, which is more typical of traditional development methodologies.

Pros of Feature Driven Development (FDD)

- **High-Level Visibility:** FDD offers high-level visibility of progress and results, which is useful for both developers and stakeholders.
- **Effective for Large Projects:** Due to its organized nature and focus on individual features, FDD is particularly effective for managing large-scale projects and coordinating with multiple teams.
- **Enhanced Predictability:** The structured approach of FDD allows for improved predictability in terms of delivery timelines.

Cons of Feature Driven Development (FDD)

- Not Ideal for Small Projects: The benefits of FDD might not be fully realized in small projects or teams, as the overhead of maintaining the process might outweigh the benefits.
- Requires Clear Definition of Features: The success of FDD largely depends on the proper definition and understanding of features. If the features are not clearly defined, it could lead to confusion and inefficiencies.
- Less Flexibility: Compared to other Agile methods, FDD has less flexibility as it requires a more structured approach and substantial planning upfront.

In conclusion, FDD is a structured Agile method that can provide the discipline and high-level visibility of traditional methods while offering the iterative benefits of Agile for developing complex, large-scale software projects. However, it might be less suitable for smaller projects and requires clear feature definitions for success.

2.3.13 Software Development Techniques: Test-Driven Development (TDD)

What is Test-Driven Development (TDD)?

Test-Driven Development (TDD) is a software development technique where the tests are written before the code. In this approach, a developer first writes an automated test case that defines a new function or improvement, then produces the minimum amount of code to pass that test, and finally refactors the new code to meet acceptable standards.

The TDD cycle involves the following steps: write a test, run all tests and see if the new one fails, write the code, run tests, refactor code, and repeat. This technique is often associated with Agile development practices like Extreme Programming (XP), but can be used in other development methodologies as well.

Why is TDD Special When Compared to Other Techniques?

TDD stands out from other methodologies because it puts testing at the forefront of the development process. Instead of writing tests after the code has been created, as is traditionally done, TDD insists on writing tests before the code is written. This helps to ensure that all code is covered by testing and allows developers to catch and correct issues early in the development process.

Pros of Test-Driven Development (TDD)

- **Better Code Quality:** TDD often results in code that is more modular and flexible, as developers need to think about the code design more carefully to make it more testable.
- **Easier Maintenance:** With TDD, you end up with a suite of tests that can be run at any time, ensuring that refactoring or adding features doesn't break existing functionality.
- **Reduces Bugs in Production:** Since the code is rigorously tested from the very beginning, the chances of encountering bugs in production are greatly reduced.

Cons of Test-Driven Development (TDD)

- Slow at the Start: Writing tests before writing code can be a slower process initially, especially for new features where the desired functionality may not be fully understood at the outset.
- Steep Learning Curve: TDD requires a different mindset and approach to coding, which can present a steep learning curve for developers accustomed to traditional coding practices.
- Incomplete Testing: TDD is not a silver bullet for catching all types of bugs. Other forms of testing (like integration testing and user acceptance testing) are still necessary.

In conclusion, TDD is a powerful methodology that can result in cleaner, less error-prone code and make the development process more streamlined. However, it does require a shift in mindset, and can initially seem slower than traditional development practices.

2.3.14 Software Development Techniques: Behavior Driven Development (BDD)

What is Behavior Driven Development (BDD)?

Behavior Driven Development (BDD) is a software development methodology that emerged from Test-Driven Development (TDD). It enhances TDD by writing test cases in a natural language that non-programmers and business stakeholders can read.

BDD focuses on the behavior of an application for the end user. It is based on the idea of developing software through examples, often referred to as 'scenarios' or 'user stories'. These scenarios guide the development and encourage collaboration between team members.

Why is BDD Special When Compared to Other Techniques?

BDD stands out because of its focus on communication and collaboration. It aims to help developers, testers, business analysts, and stakeholders all speak the same language and understand the product from the user's perspective. The idea is to provide a single source of truth for everyone involved in the development process.

By defining behavior in terms of user stories, it also ensures that the developed features directly align with customer requirements and the software provides value to the user.

Pros of Behavior Driven Development (BDD)

- Improved Collaboration: BDD improves collaboration between tech and non-tech team members by using simple language to describe system behavior. This ensures a better understanding of the system among all stakeholders.
- Customer Focused: BDD helps to ensure that the software developed is directly in line with the needs and expectations of the customer.
- Regression Protection: As with TDD, BDD also results in a suite of tests that can be run at any time to check if new changes break any existing functionality.

Cons of Behavior Driven Development (BDD)

- Requires Strong Communication: The success of BDD heavily depends on the effective communication of requirements and expectations among stakeholders, developers, and testers.
- Time Consuming: Writing user stories and scenarios in a way that accurately and fully describes each feature can be time-consuming.
- Steep Learning Curve: Like TDD, BDD can be difficult to implement if the team is not used to writing tests first or thinking from the user's perspective.

In conclusion, BDD is a powerful methodology that brings together the technical and non-technical members of a team, focusing development on delivering features that provide the most value to the customer. It's not without its challenges, but can provide immense benefits when implemented properly.

2.3.15 Comparison differences between each Software Development Techniques

Table 2. 2 Comparison differences between each Software Development Techniques

Software Development Techniques	Key Features	Strengths	Weaknesses
Waterfall Model	Sequential, each phase completed before moving to the next.	Simple and easy to understand, well-documented. Rigid design and save time and money.	Difficult to make changes, inflexible. Customers are unable to make changes during the developments.
Prototype Methodology	Create a simplified model before the final product	Allows for early feedback and reduces risk. High customer involvement. Both parties have a clear idea.	Can be time-consuming and may lead to insufficient analysis. Both parties should have knowledge of prototyping.
Spiral Model	Iterative, focuses on risk management.	Allows for changes, good for large projects. Have good risk management.	Complex, requires significant risk analysis. Require high cost hence not appropriate for low budget projects.
Agile	Iterative, flexible, customer-focused.	Adaptable to changes, customer-	Can be difficult to estimate, requires

		<p>centric, promotes team collaboration.</p> <p>Can directly communicate with the customer when the project is ongoing.</p> <p>Can make changes while project in ongoing.</p>	<p>customer involvement.</p> <p>Lack of documentation.</p> <p>The customer should have sound knowledge of the project going on.</p>
Scrum	<p>Subset of Agile, uses Sprints for development.</p>	<p>High productivity, allows for changes.</p> <p>Have a scrum master to look on the project.</p>	<p>Requires experienced team members, not suitable for all project types.</p> <p>Need experienced team.</p> <p>Suitable for small projects.</p> <p>Due to continuous updating and reporting the scrum master's time could be wasted.</p>
Extreme Programming (XP)	Agile-based, focuses on code quality.	Enhances code quality, promotes teamwork.	High level of discipline required,

			can be overkill for simple projects.
Lean Software Development	Agile-based, focuses on efficiency.	Eliminates waste, delivers fast. Suitable for low budget projects. Able to deliver product early.	Requires a culture shift, can be difficult to implement. Business Analyst may decide the success.
Rapid Application Development (RAD)	Iterative, quick development.	Speeds up development process, flexible. Requirements are prioritized. Suitable for large projects.	Not suitable for small projects, requires strong team. Require skilled team. Not appropriate for small budget projects.
Joint Application Development (JAD)	Collaborative, involves stakeholders.	Encourages communication, involves user feedback. Can have off site conferences. Business Analysts can collect requirements while in the development process.	Requires commitment from stakeholders, may lead to scope creep. Business Analyst may decide the success. Require skilled team.

			May need more time to planning.
Dynamic Systems Development Method (DSDM)	Agile-based, focuses on frequent releases.	Delivers quickly, encourages user involvement.	Requires skilled teams, can be expensive.
Feature Driven Development (FDD)	Agile-based, focuses on features.	Delivers tangible results, scalable.	Requires detailed documentation, less communication.
Test-Driven Development (TDD)	Test cases written before development.	Improves code quality, reduces bug count.	Can be time-consuming, requires experience.
Behavior Driven Development (BDD)	Extension of TDD, focuses on behavior.	Enhances collaboration, clarifies requirements.	Requires strong communication, may be difficult to implement.

2.3.16 Chosen Development Technique for BAUHINIA Project

Chosen Development Technique for BAUHINIA Project is Agile Scrum. There are several reasons why Agile Scrum would be a beneficial development technique for the BAUHINIA project:

1. Iterative Development:

Scrum, as a subset of Agile, emphasizes an iterative and incremental approach to software development. This approach breaks the project down into manageable chunks (known as "sprints") which typically last between one and four weeks. This would allow the BAUHINIA project team to focus on a few deliverables at a time, quickly adapt to changes, and continuously improve the product throughout its development.

2. Flexibility:

Scrum allows for flexibility and adaptability. Unlike the Waterfall model, where requirements are expected to be fully defined at the outset, Scrum welcomes changes and enhancements throughout the project. This would be particularly beneficial for the BAUHINIA project if the project requirements are likely to evolve or if the team is working in a rapidly-changing environment.

3. Collaboration and Communication:

Scrum promotes close collaboration, regular communication, and transparency among team members. Daily stand-up meetings (called "Daily Scrums") keep everyone informed about what's been done, what's in progress, and any obstacles that have arisen. This helps to identify issues early on, leading to quicker resolution, and would help ensure a smooth development process for the BAUHINIA project.

4. Customer Satisfaction:

In Scrum, working software is delivered to customers at the end of each sprint, enabling them to provide feedback and input that can be incorporated in the next sprint. This frequent interaction and feedback loop would help ensure the BAUHINIA project aligns with customer expectations and needs, ultimately enhancing customer satisfaction.

5. Adaptive planning:

Scrum follows an iterative approach to software development, allowing changes to be incorporated throughout the project development lifecycle. This is ideal for the BAUHINIA project as it may require adjustments to requirements, design, and implementation strategies as the project progresses.

6. Incremental delivery of software:

Scrum promotes the development of software in small manageable units, with each iteration or sprint resulting in a potentially shippable increment of work. This incremental approach allows for faster user feedback and the ability to improve or pivot as necessary.

7. Enhanced team collaboration and communication:

The Scrum framework facilitates regular communication among the project team and stakeholders, fostering an environment of transparency and collaboration. Daily scrum meetings, sprint planning, reviews, and retrospectives ensure that all team members are on the same page and any issues are identified and addressed promptly.

8. Risk reduction:

Due to its iterative nature, risks and issues are identified and addressed more quickly in Scrum. Regular feedback loops with the product owner and stakeholders help in catching

misalignments early and adjusting the course of the project, reducing the project overall risk.

Comparison with Other Techniques:

While other methods like Extreme Programming (XP) and Lean Software Development also support iterative development and could potentially be used for the BAUHINIA project, Scrum is chosen for its explicit emphasis on flexibility, regular communication, and customer involvement. On the other hand, traditional methods like the Waterfall Model or Prototype Methodology, which involve more rigid and linear processes, might not offer the same level of adaptability and responsiveness to changes that Scrum can provide.

In comparison to other techniques:

- Waterfall Model:

The Waterfall Model is a linear sequential design approach, which may not provide the flexibility needed for the BAUHINIA project.

- Prototype Methodology, Spiral Model, Rapid Application Development (RAD), and Joint Application Development (JAD):

While these models encourage user feedback, they might not be as well-suited for managing changes and do not promote the continuous integration of working software like Scrum.

- Extreme Programming (XP):

While XP is also an Agile methodology and emphasizes customer satisfaction, Scrum's focus on self-organizing teams and the division of work into sprints can be more beneficial for managing the BAUHINIA project.

- Lean Software Development:

While Lean provides a set of principles that can be beneficial in many contexts, Scrum offers a more concrete set of practices and roles which may be easier to adopt for BAUHINIA project.

- Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), Test-Driven Development (TDD), Behavior Driven Development (BDD):

These methodologies each have their own strengths and are often used in conjunction with Scrum. However, they either focus more narrowly on certain aspects of development (like testing or features), or are more complex and less widely understood and adopted than Scrum.

Overall, for a project like BAUHINIA, which presumably values adaptability, speed, collaboration, and user feedback, Agile Scrum would be an excellent choice.

2.4 Conclusion

In conclusion, the comprehensive exploration of software development methodologies and tools has revealed the strengths, weaknesses, and ideal application scenarios for each. Here's a brief recap of my findings:

Software Development Methodologies: From Waterfall to Agile Scrum, to Behavior Driven Development (BDD), each methodology offers a unique approach to software development. While Waterfall may be ideal for well-defined projects with static requirements, Agile methodologies, like Scrum, bring flexibility and adaptability, making them better suited for projects with evolving requirements.

The Agile Scrum method was found to be especially advantageous due to its iterative nature, clear communication channels, incremental delivery approach, and effective risk mitigation, making it an excellent choice for the BAUHINIA project.

Software Development Tools: The evaluation of various software development tools demonstrated their value in aiding software development processes. Diagram tools can simplify the understanding of complex systems, project management tools can streamline workflows, and programming tools can accelerate coding. Similarly, web development tools, configuration management tools, quality assurance tools, documentation tools, design tools, maintenance tools, analysis tools, process modeling tools, prototyping tools, and change control tools each serve unique and critical roles.

For the BAUHINIA project, a combination of these tools would likely be beneficial. For instance, project management tools can aid in task assignment and tracking, programming tools can support coding tasks, while quality assurance tools can help ensure the software meets its quality criteria.

Ultimately, the chosen methodologies and tools should be tailored to the unique demands of the BAUHINIA project. The Agile Scrum methodology was found to be most suitable due to its flexible and adaptive nature. In terms of tools, the specific ones to be used should be based on the project's specific requirements and the team's expertise.

The conclusion of this investigation reaffirms the importance of carefully evaluating and selecting software development methodologies and tools in relation to project specifics, team capabilities, and project environment. This meticulous selection process can greatly enhance the project's chances of success.

Activity 3

3.1 BAUHINIA Project Business Application Review

3.1.1 Presentation



Figure 3. 1 Slide 1

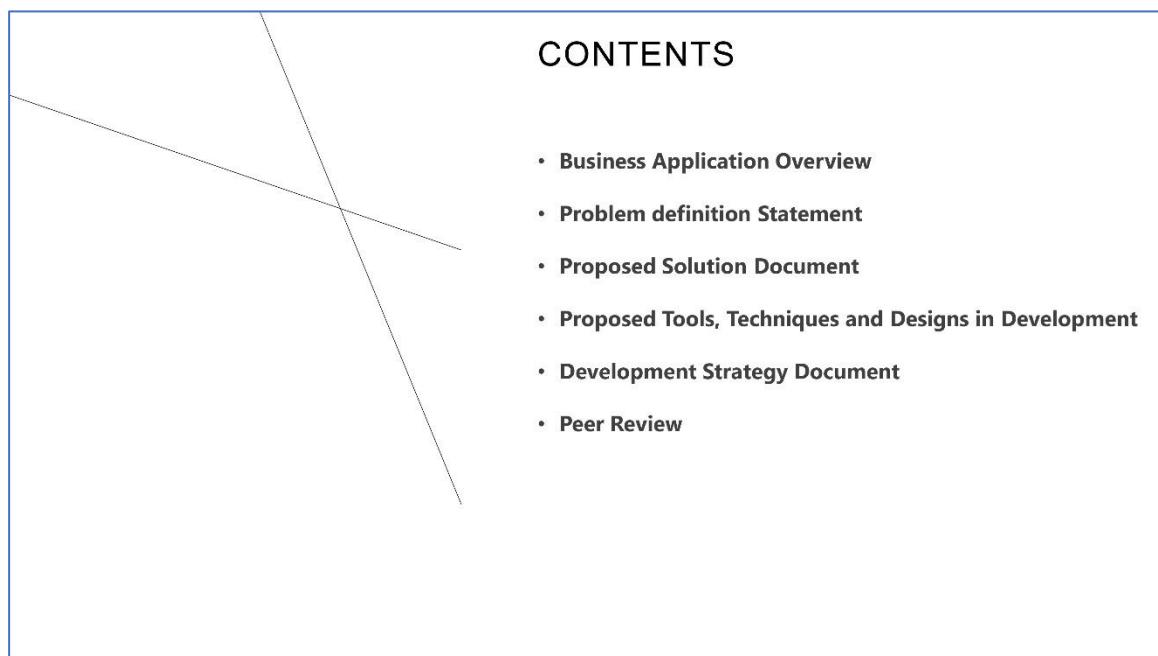


Figure 3. 2 Slide 2

BUSINESS APPLICATION OVERVIEW

Introduction to BAUHINIA Project

- The BAUHINIA project aims to revolutionize the way BAUHINIA Clothing handles its business operations.
- By moving from a social-media-based ordering system to a full-fledged online solution, BAUHINIA is set to improve customer engagement and streamline their inventory management.
- The BAUHINIA Project is an initiative by AKL Software to address the operational challenges of BAUHINIA, a popular clothing brand in Sri Lanka.
- Moving away from the constraints of social-media-based ordering, the project is an endeavor to leverage the power of agile software development and create an online solution.

Figure 3. 3 Slide 3

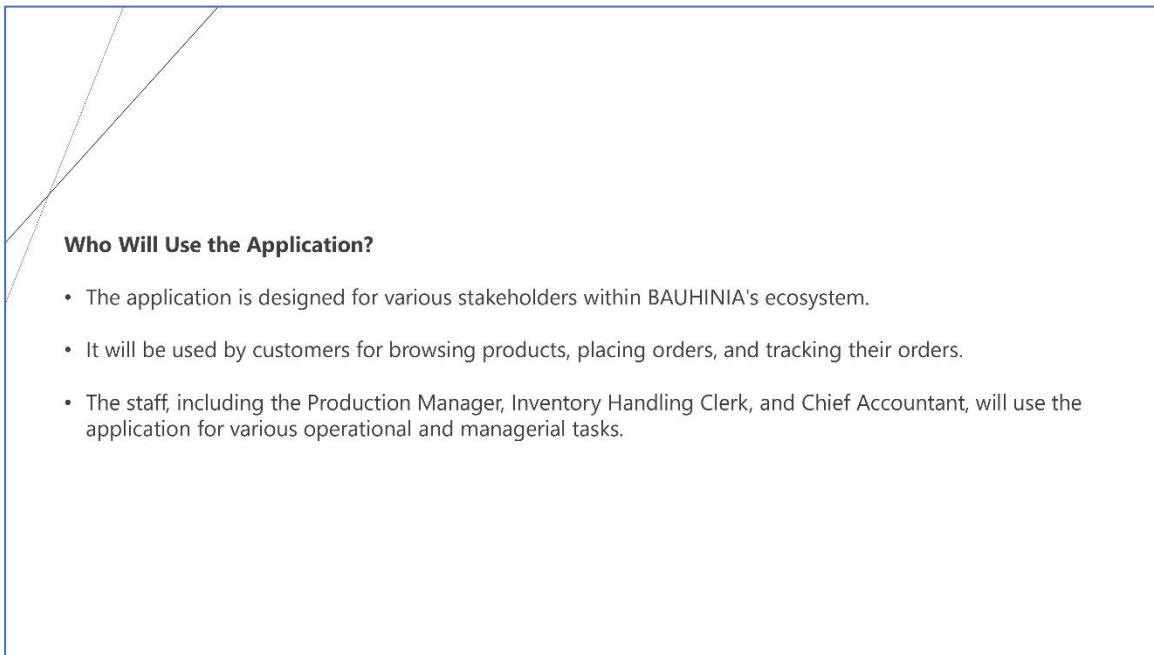
The Need for the New Business Application

- BAUHINIA is growing at a rapid pace, and the old approach of handling orders and managing inventory through social media networks is increasingly cumbersome.
- There is a pressing need for a robust, user-friendly business application that can streamline various operational tasks, enhance customer engagement, and improve the overall efficiency of the business.

Primary Objectives of the Business Application

- The new business application aims to automate BAUHINIA's workflow and eliminate manual paperwork associated with inventory management.
- The application seeks to provide an organized platform for tracking orders, managing inventory, creating daily and monthly reports, and improving overall business operations.

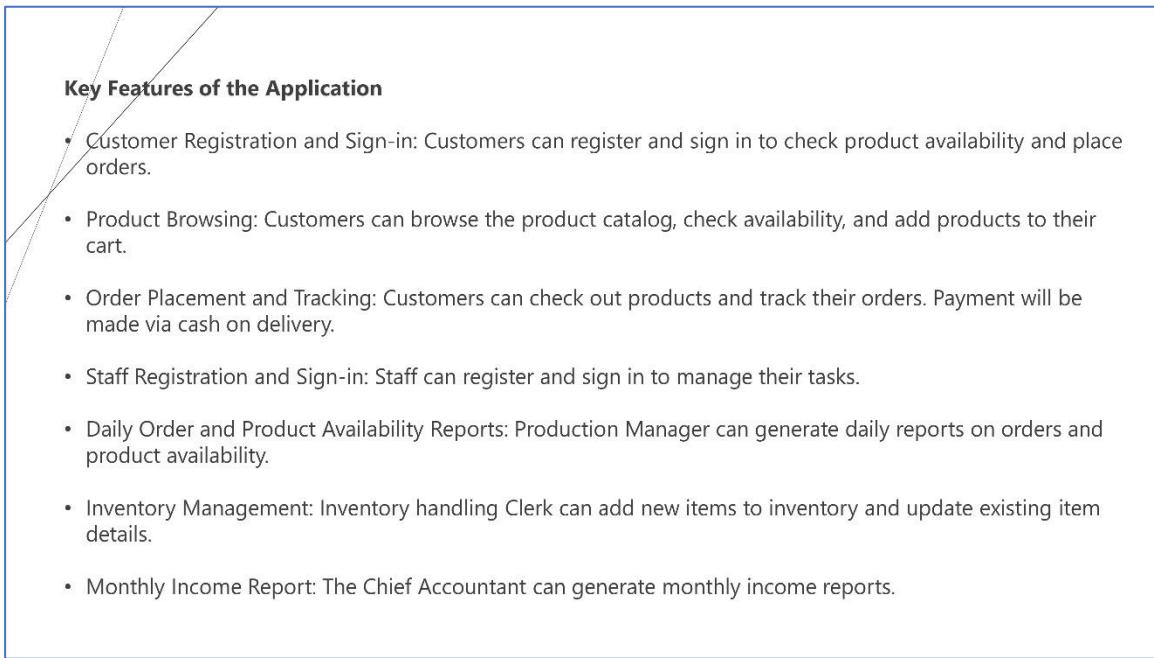
Figure 3. 4 Slide 4

A decorative background graphic in the top-left corner of the slide, consisting of several thin, light-grey lines that intersect and form a triangular shape.

Who Will Use the Application?

- The application is designed for various stakeholders within BAUHINIA's ecosystem.
- It will be used by customers for browsing products, placing orders, and tracking their orders.
- The staff, including the Production Manager, Inventory Handling Clerk, and Chief Accountant, will use the application for various operational and managerial tasks.

Figure 3. 5 Slide 5

A decorative background graphic in the top-left corner of the slide, consisting of several thin, light-grey lines that intersect and form a triangular shape.

Key Features of the Application

- Customer Registration and Sign-in: Customers can register and sign in to check product availability and place orders.
- Product Browsing: Customers can browse the product catalog, check availability, and add products to their cart.
- Order Placement and Tracking: Customers can check out products and track their orders. Payment will be made via cash on delivery.
- Staff Registration and Sign-in: Staff can register and sign in to manage their tasks.
- Daily Order and Product Availability Reports: Production Manager can generate daily reports on orders and product availability.
- Inventory Management: Inventory handling Clerk can add new items to inventory and update existing item details.
- Monthly Income Report: The Chief Accountant can generate monthly income reports.

Figure 3. 6 Slide 6

Benefits of the Business Application

- Improved Efficiency: By automating various operational tasks, the application will significantly reduce the manual workload and improve efficiency.
- Enhanced Customer Engagement: The application will provide a user-friendly platform for customers to engage with BAUHINIA, leading to improved customer satisfaction and loyalty.
- Streamlined Inventory Management: The application will make inventory management a breeze, reducing errors and ensuring optimal stock levels.

Technology and Security of the Business Application

- The application will be built using cutting-edge technology to ensure smooth functionality, scalability, and a seamless user experience.
- Emphasis will be placed on data security, with advanced encryption methods implemented to safeguard customer and business data.

Figure 3. 7 Slide 7

Roles and Responsibilities within the Application

- Different staff members will have different roles within the application. This includes creating and updating inventory records, generating daily orders and product availability reports, and creating monthly income reports.
- Customers will have a secure, personalized space to browse products, place orders, and track their purchases.

Usability and Accessibility of the Application

- The business application will be designed with a focus on usability and accessibility, ensuring that it provides a user-friendly experience for both staff members and customers.
- It will be accessible across multiple devices, including smartphones, tablets, and computers, allowing users to engage with the application anytime, anywhere.

Figure 3. 8 Slide 8

Future Scope of the Application

- The business application is not a static solution but will be subject to continual improvements and updates to better serve the needs of BAUHINIA.
- Future enhancements may include integration with other platforms, implementation of AI for demand forecasting, and adding new features based on user feedback and business requirements.

Summary of Business Application Overview

- The BAUHINIA Project is a bold step forward in streamlining the business operations of the BAUHINIA clothing brand.
- The new business application aims to address the brand's operational challenges, provide an enhanced shopping experience for customers, and create a more efficient and productive working environment for staff members.
- This application represents an important part of BAUHINIA's digital transformation, with the potential to drive significant growth and success for the brand in the coming years.

Figure 3. 9 Slide 9

PROBLEM DEFINITION STATEMENT

Problem Statement

- BAUHINIA is struggling with inefficiencies in order processing and inventory management due to the manual nature of these processes.
- Increased order volume from growing popularity has overwhelmed the current system.
- The manual system is unscalable, time-consuming, labor-intensive, and prone to errors, leading to potential dissatisfaction and harm to BAUHINIA's reputation.

Figure 3. 10 Slide 10

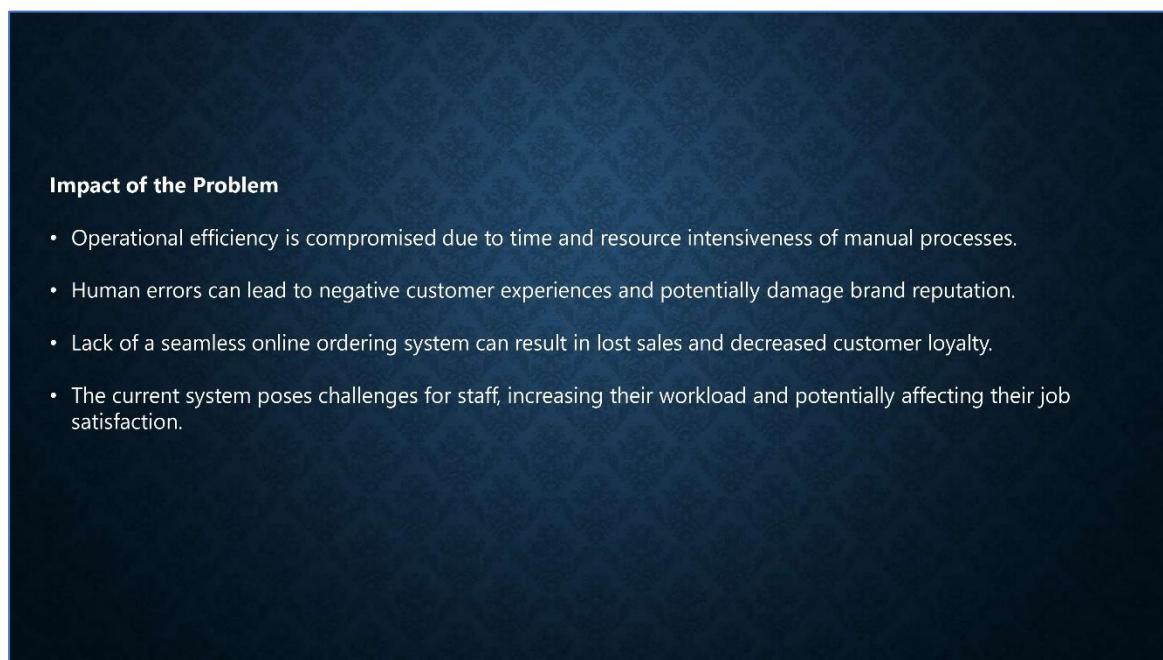


Figure 3. 11 Slide 11

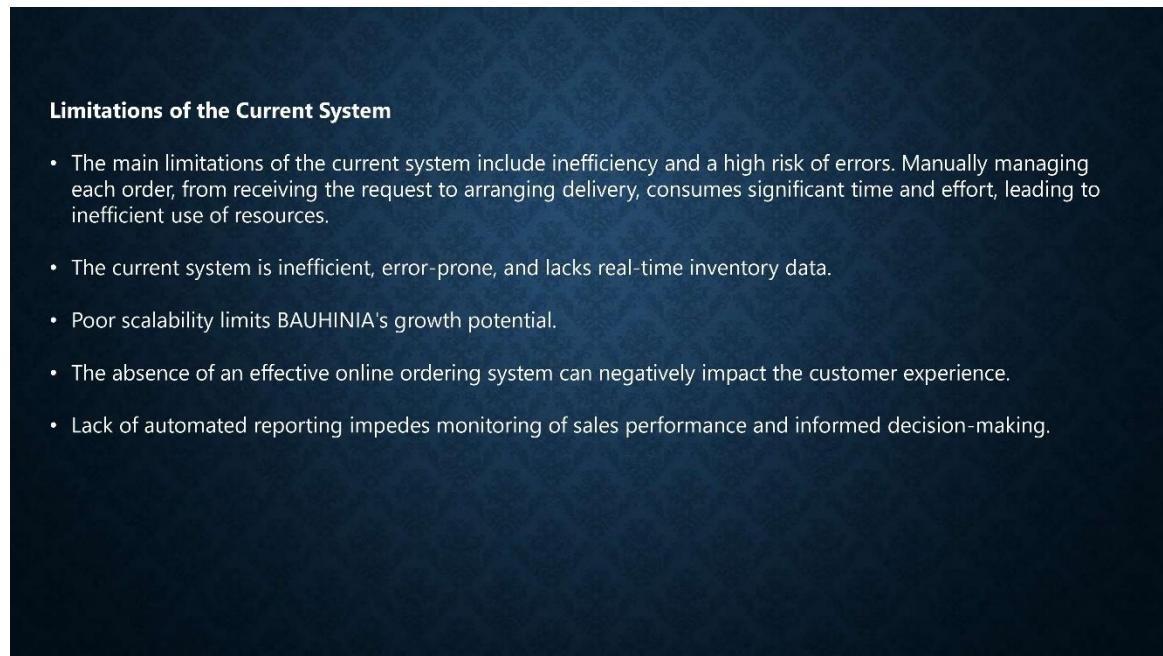


Figure 3. 12 Slide 12

Desired Outcomes from the New Solution

- The new solution should improve operational efficiency, reduce errors, and provide real-time inventory tracking.
- It should enhance scalability, customer experience, and enable automated reporting.
- The system should also allow secure user registration and sign-in, with flexible access levels for staff members.

Risk Areas in Developing the New System

- Potential risks include technical complexity, data migration issues, system security concerns, user acceptance, cost overruns, and dependency on the vendor.
- These risks should be proactively managed to ensure the successful implementation of the new system.

Figure 3. 13 Slide 13

PROPOSED SOLUTION DOCUMENT

Proposed Solution – Introduction

- BAUHINIA Inventory Control Application aims to address the operational inefficiencies present in the current system.
- The new system will include features like product browsing, product details, shopping cart, checkout process, order tracking for customers, and login, sign-up, reports, inventory update, and a dashboard for staff.

Figure 3. 14 Slide 14

Limitations of the Existing System

- Limited Accessibility: Customers must physically/order through social media visit the store for purchases, limiting reach.
- Inefficient Inventory Management: The manual system may result in recording errors, stock mismanagement, and difficulties in tracking real-time inventory levels.
- Limited Customer Reach: Operating solely social media based offline restricts the customer base to the local area.
- Manual Record Keeping: Maintaining manual records can be time-consuming and prone to human errors. Analyzing data for insights is also challenging.
- Lack of Data Analysis: Due to the absence of digitized and structured data, it's difficult to analyze sales trends, customer behavior, and inventory management.
- Absence of Customer Profile: Without an online platform, BAUHINIA cannot create customer profiles for personalizing the shopping experience and improving customer loyalty.

Figure 3. 15 Slide 15

Proposed System – Objectives

- The goal is to transition BAUHINIA from an social media based offline manual system to an automated, online platform.
- The proposed system will provide customers with the convenience of shopping from anywhere, at any time.
- It aims to enhance the shopping experience, manage inventory efficiently, reach a broader customer base, and enable data-driven strategic business decisions.

Proposed System Analysis – Introduction

- The proposed system for BAUHINIA is an e-commerce solution designed to transition BAUHINIA's current manual operations to an automated online platform.
- The system aims to increase operational efficiency, expand customer reach, and enhance the shopping experience.

Figure 3. 16 Slide 16

Proposed System – Key Features

The new system will include features like product browsing, product details, shopping cart, checkout process, order tracking for customers, and login, sign-up, reports, inventory update, and a dashboard for staff.

- Customer Interface: Features like account creation, login, product browsing, product details, shopping cart, checkout, and order tracking will be available.
- Staff Interface: Features like account creation, login, report generation, inventory updating, and a dashboard for an overview of business operations will be available.
- Product Management: An extensive product database for easy management and updating by staff.
- Shopping Cart and Checkout: A seamless and secure process for customers.
- Order Management and Tracking: Efficient order management by staff and easy order tracking for customers.
- Report Generation: Functionality to generate various reports for informed business decision-making.
- Security and Compliance: Adherence to security standards and data protection laws to ensure safe transactions and protect customer data.

Figure 3. 17 Slide 17

Benefits of the Proposed System

- The proposed system aims to overcome the limitations of the existing system and provide a scalable solution to support BAUHINIA's business growth.
- By transitioning to this system, BAUHINIA can enhance the shopping experience for its customers, manage operations more efficiently, and leverage data for strategic decision making.

Figure 3. 18 Slide 18

PROPOSED TOOLS, TECHNIQUES AND DESIGNS IN DEVELOPMENT

Proposed Solution – Introduction

- The project will be handled using Agile Scrum methodology, a collaborative and iterative approach to software development that allows for flexibility and rapid adjustments throughout the process.
- Diagrams created for the project include ERD Diagram, Class Diagram, Use Case Diagram, Sequence Diagrams, User Interfaces Design, and Logical Database Diagram Design. These diagrams help to illustrate the system's design, its components, and how they interact.

Figure 3. 19 Slide 19

Diagramming Tools

- Microsoft Visio and Draw.io have been chosen as the primary diagramming tools for creating flowcharts, process diagrams, and other graphical representations needed in this project.

Programming Tools

- PyCharm and Visual Studio Code, powerful and versatile Integrated Development Environments (IDEs), will be used for writing and testing the application's code.

Web Development Tool

- Django, a high-level Python web framework, will be used to build the application. Django promotes rapid development and clean, pragmatic design, perfect for a project of this scale.

Figure 3. 20 Slide 20

Configuration Management Tools

- Git, a distributed version control system, will be used for source code management, enabling team members to work simultaneously without overwriting each other's changes.
- Ansible, an open-source automation tool, will be used for deploying the application and managing its configuration.

Quality Assurance Tools

- JUnit, a unit testing framework for the Java programming language, will help ensure the reliability and correctness of the new system's code.
- Selenium, a tool for automating web browsers, will be used for testing the web application's functionality and ensuring it behaves as expected.

Figure 3. 21 Slide 21

Documentation Tools

- Microsoft Word and Confluence will be used for project documentation, recording everything from project requirements to test cases and results.
- Markdown will be used for maintaining documentation alongside the codebase, making it easier to understand the code's purpose and functionality.

Design Tools

- Figma and Adobe XD will be used to create the application's user interface and experience designs, helping to ensure the application is user-friendly and visually appealing.

Maintenance Tools

- Jira, an issue and project tracking tool, and GitHub, a web-based hosting service for version control, will be used for tracking bugs, managing tasks, and maintaining the application's codebase.

Figure 3. 22 Slide 22

Analysis Tools

- Tableau, RapidMiner, and Google Analytics will be used for data analysis, helping the BAUHINIA team to understand their business better, make data-driven decisions, and improve their operations.

Process Modelling Tools

- Bizagi and Microsoft Visio will be used for process modelling, helping to visualize the project's workflow and understand how different processes interconnect.

Prototyping Tools

- Figma and Balsamiq will be used for creating functional prototypes of the application. These prototypes allow for user testing and feedback before the development phase.

Figure 3. 23 Slide 23

Development Technique

The Agile Scrum development technique has been selected for the BAUHINIA project. This method values flexibility, customer satisfaction, and collaborative problem solving, making it well-suited for a project of this nature.

SCRUM PROCESS

The diagram illustrates the Scrum process. It starts with a 'VISION' represented by a lightbulb icon. This leads to 'USER STORIES', depicted as blue document icons. These stories are then organized into a 'SPRINT BACKLOG', shown as orange bars. The process then enters a iterative cycle represented by a large red gear labeled '2-4 WEEKS'. Inside this cycle, a 'SCRUM MASTER' (represented by a person icon) oversees the work. A 'DAILY STAND UP' meeting (represented by three people icons) occurs every '24 HOURS'. The final output of each iteration is a 'PRODUCT INCREMENT' (represented by a grid of colored squares).

Figure 3. 24 Slide 24

Diagrams

A series of diagrams have been developed, including the ERD Diagram, Class Diagram, Use Case Diagram, Sequence Diagrams, and Logical Database Diagram Design. These diagrams will provide a visual representation of the application and its processes, aiding both development and understanding of the system.

Figure 3. 25 Slide 25

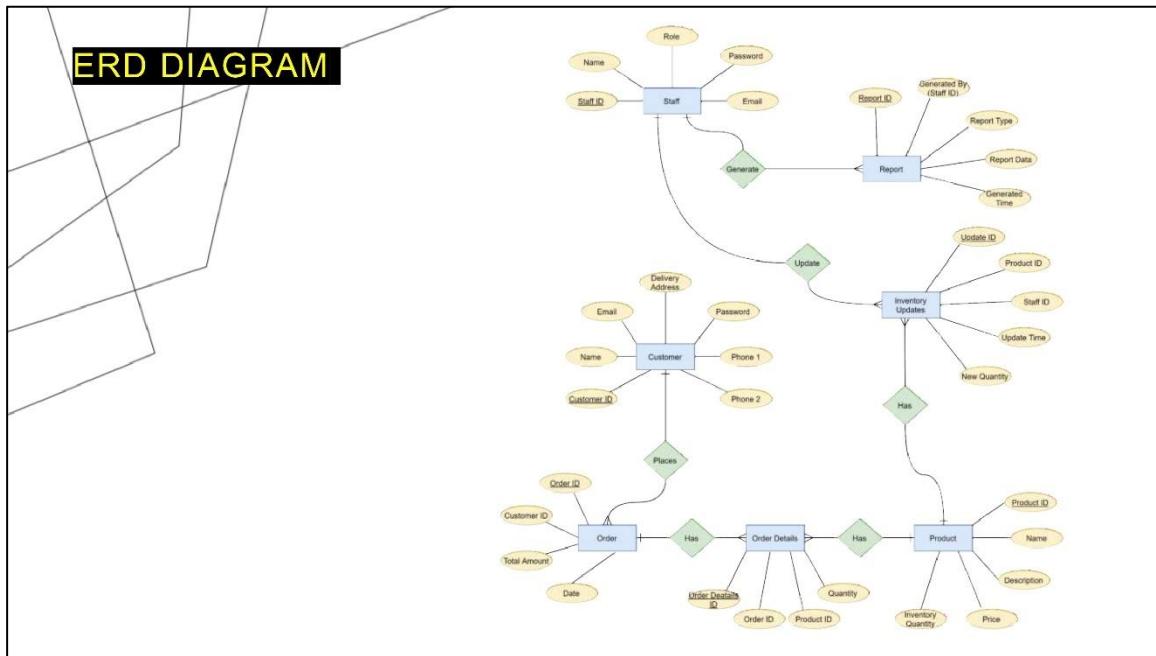


Figure 3. 26 Slide 26

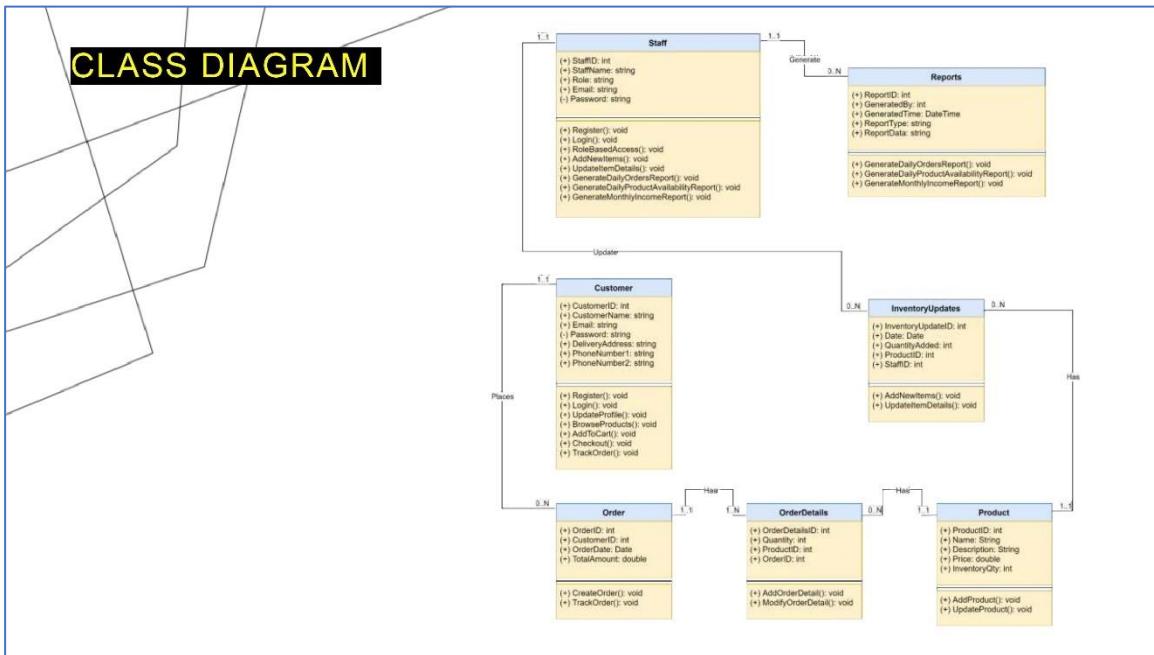


Figure 3. 27 Slide 27

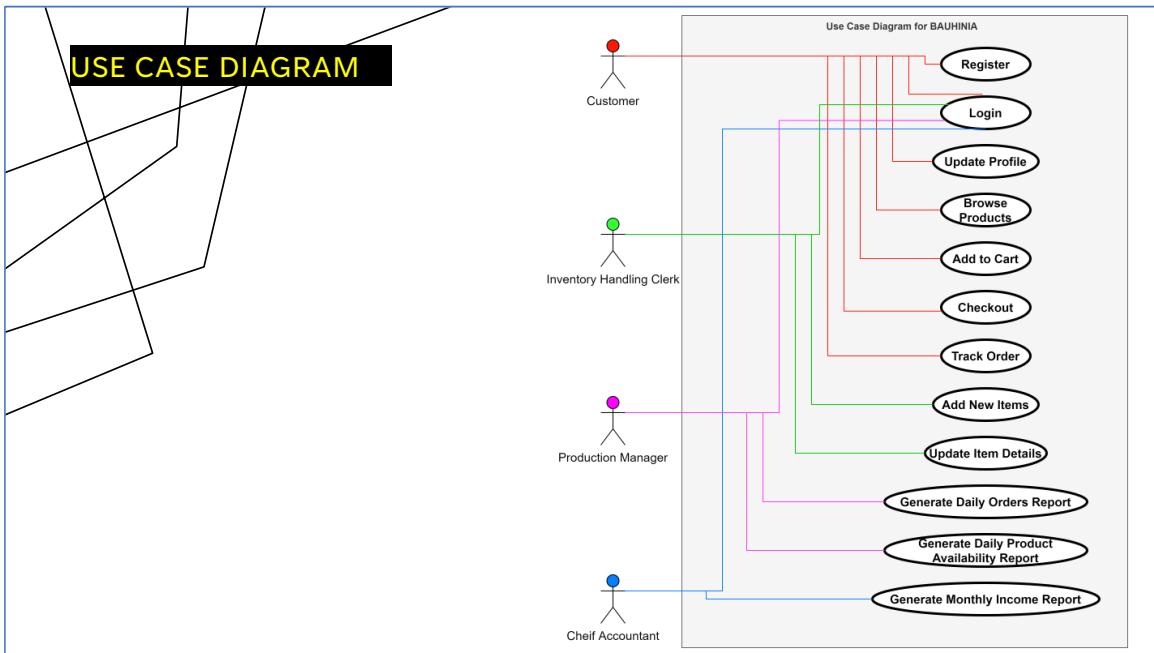


Figure 3. 28 Slide 28

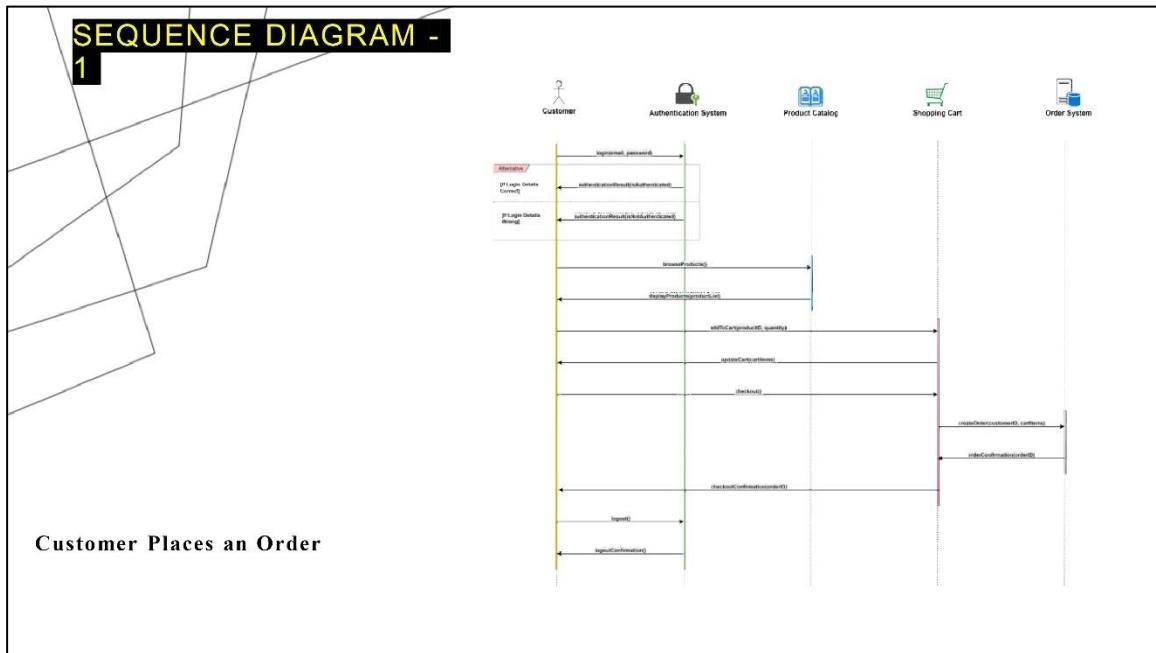


Figure 3. 29 Slide 29

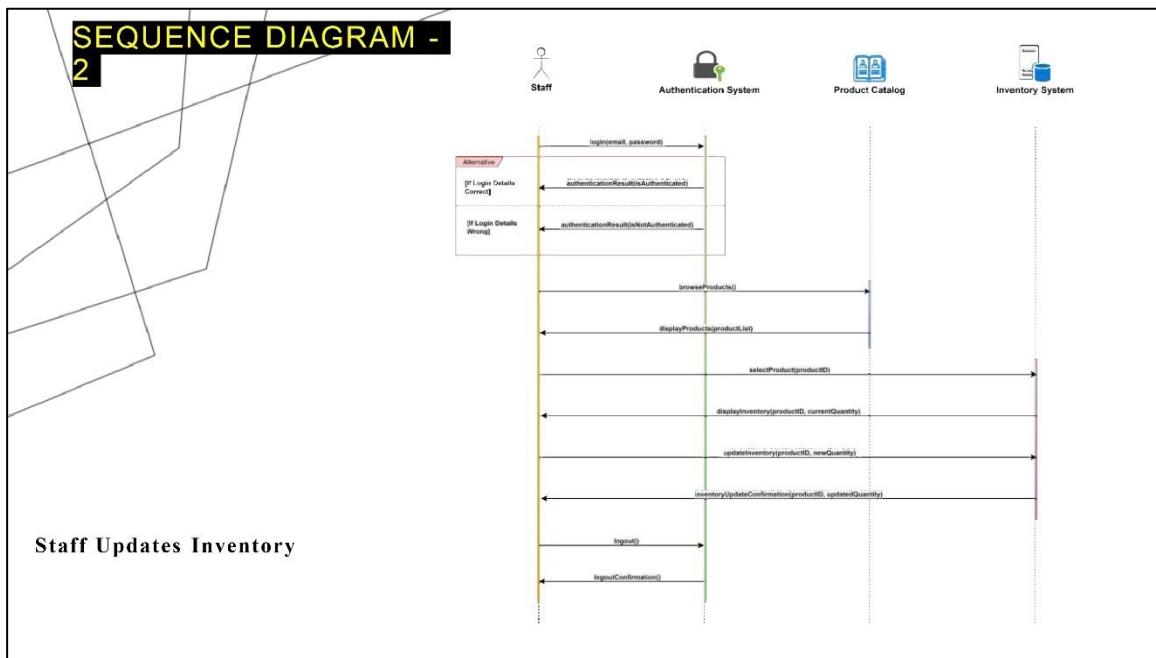


Figure 3. 30 Slide 30

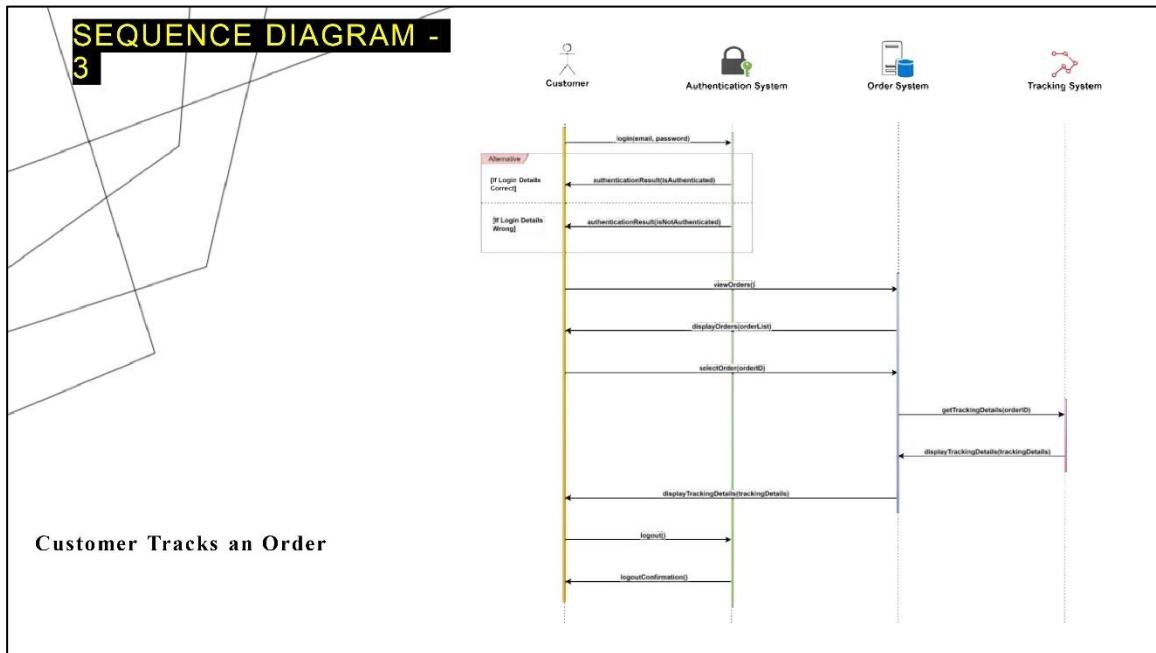


Figure 3. 31 Slide 31

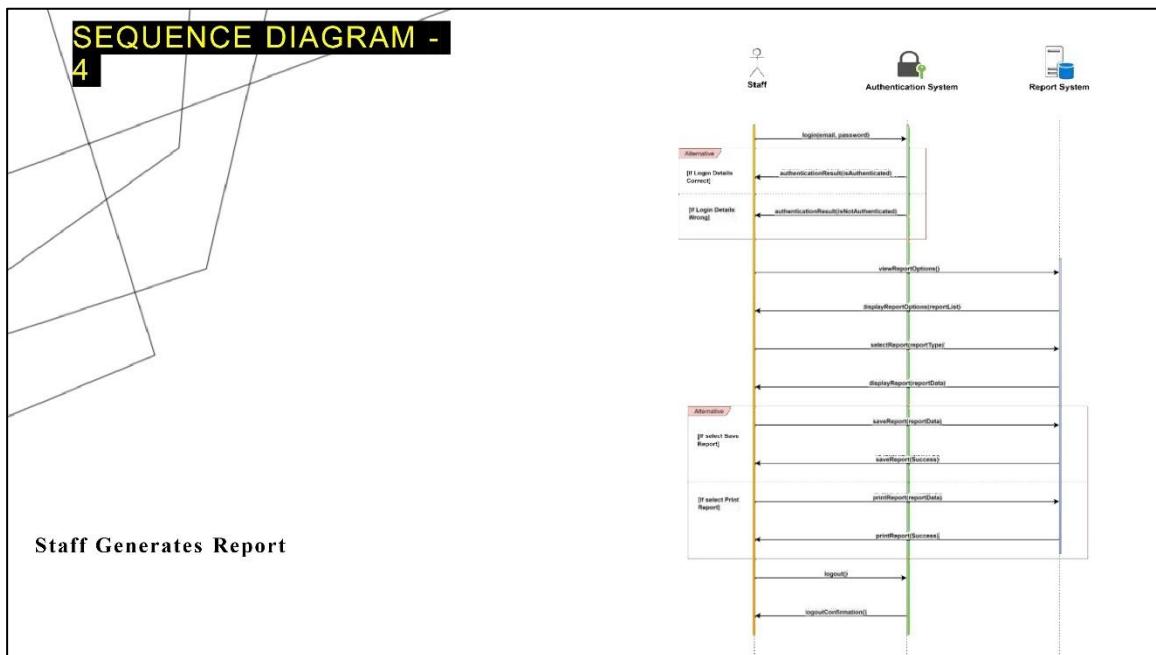


Figure 3. 32 Slide 32

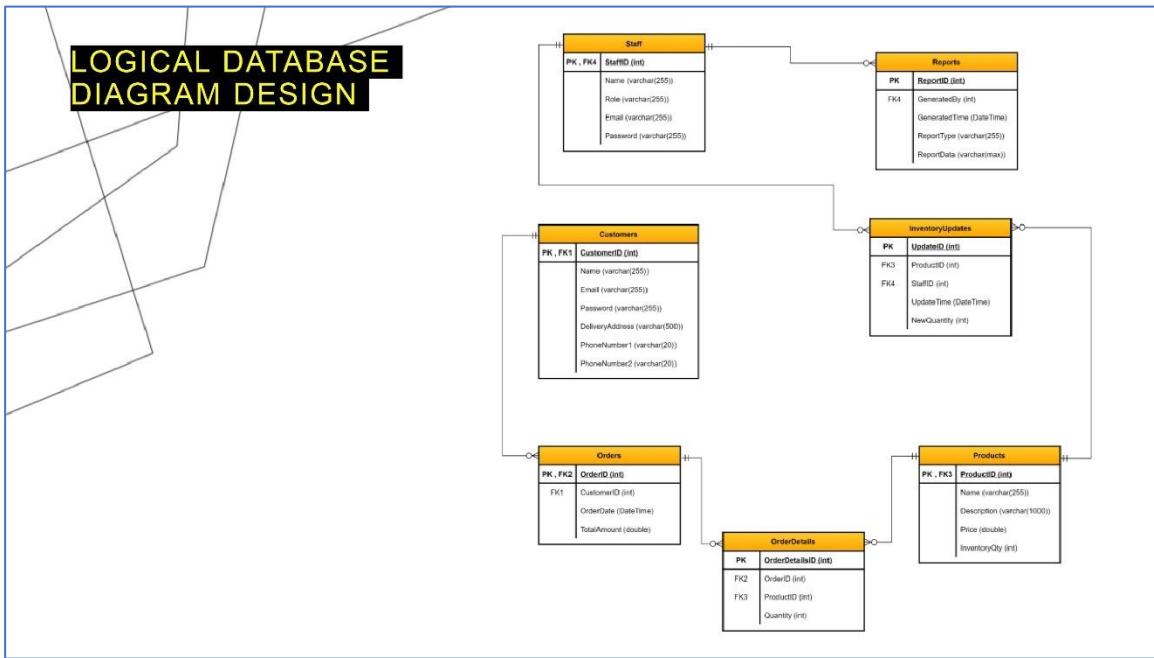


Figure 3. 33 Slide 33

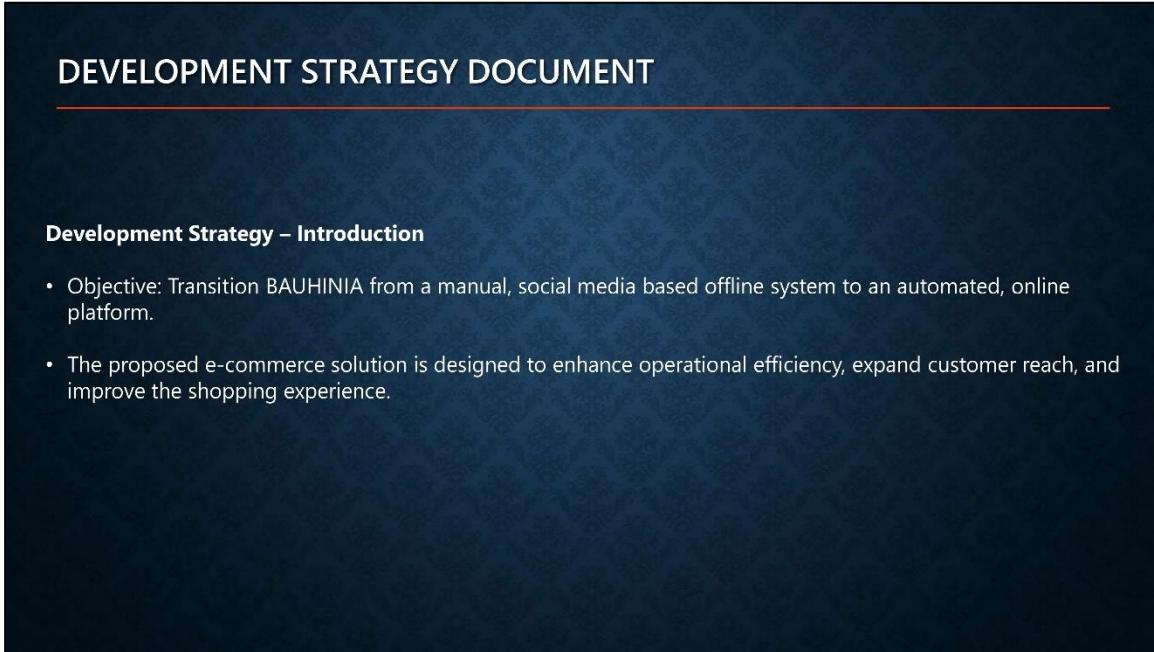


Figure 3. 34 Slide 34

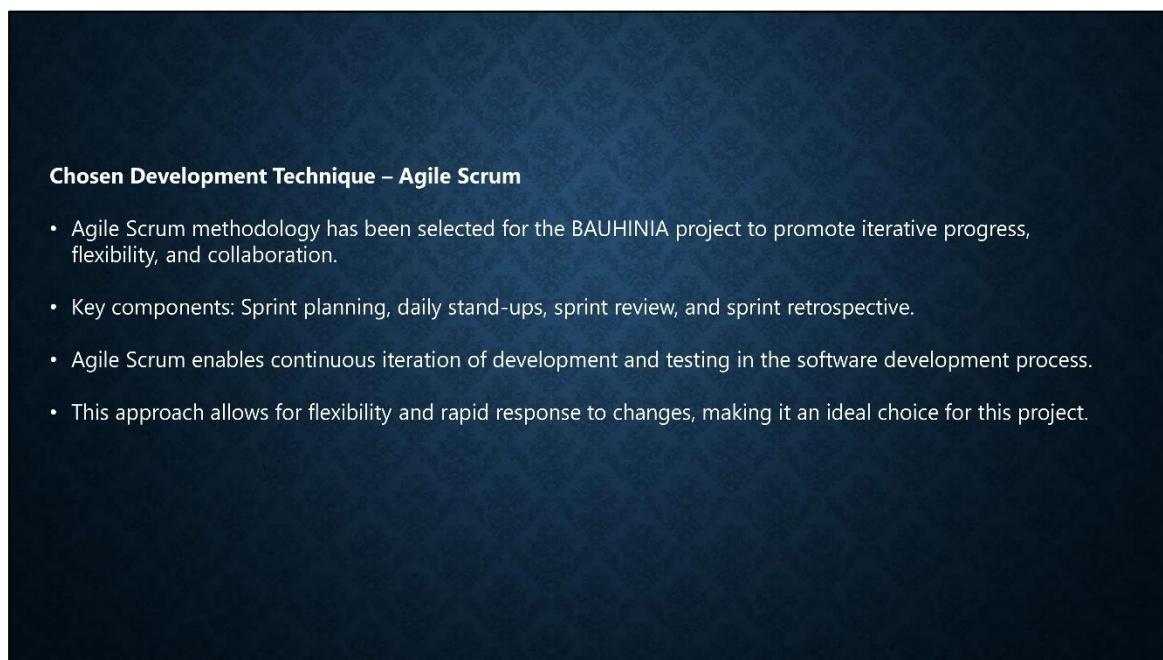


Figure 3. 35 Slide 35

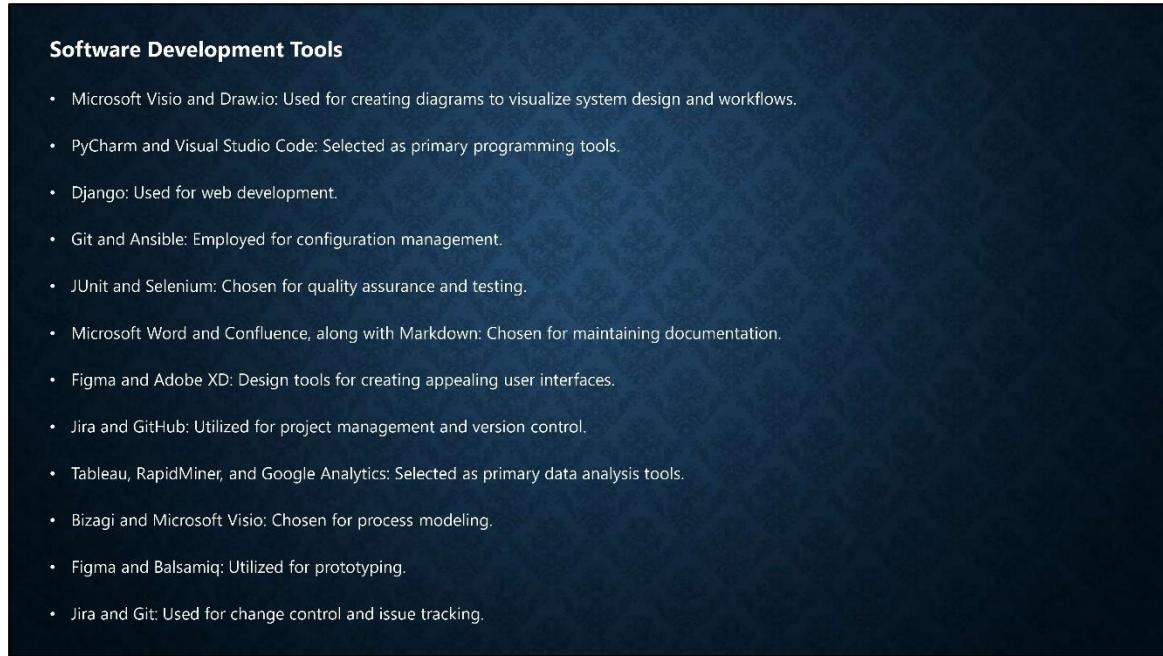


Figure 3. 36 Slide 36

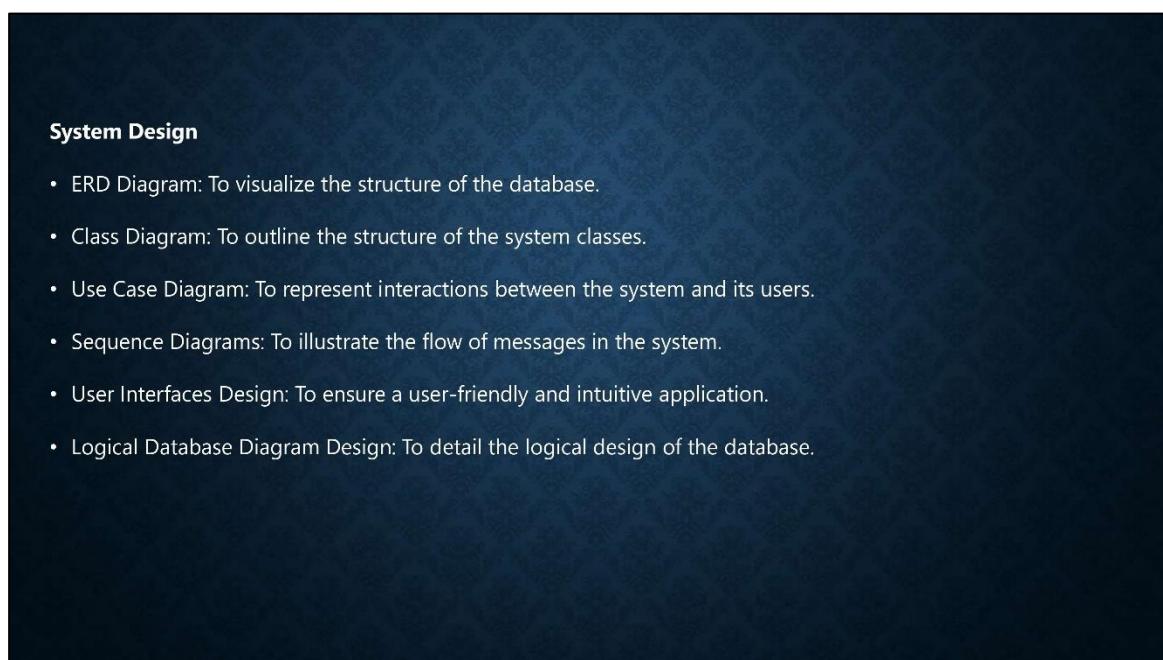


Figure 3. 37 Slide 37

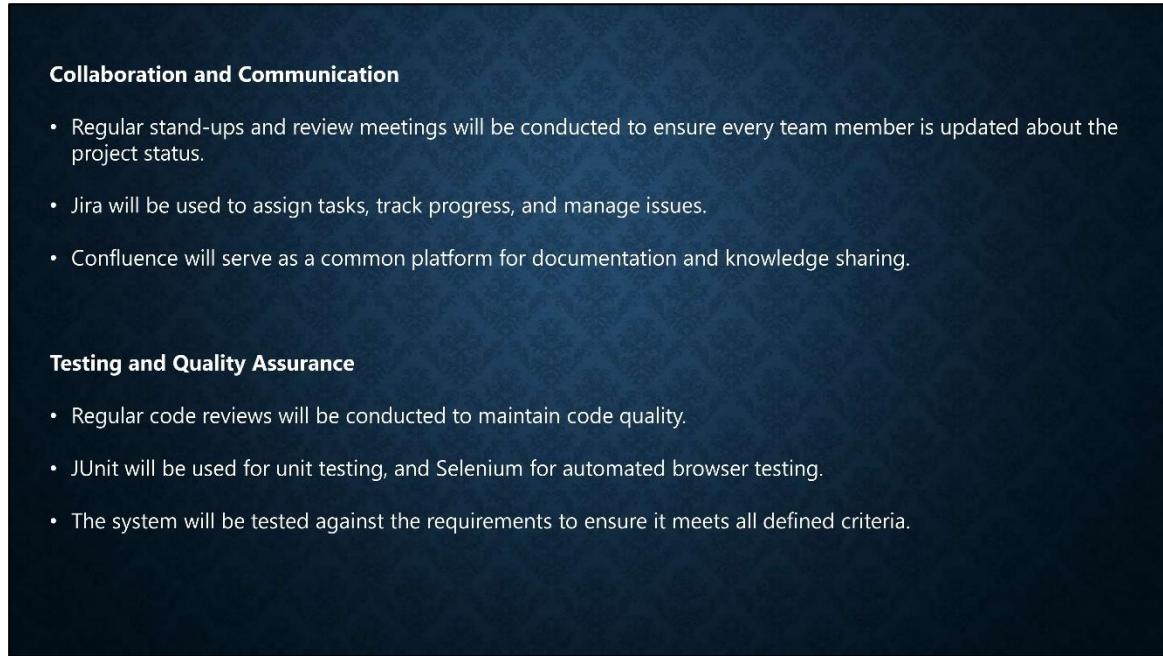


Figure 3. 38 Slide 38

Deployment and Maintenance

- Git will be used for version control, and Ansible for automating deployment processes.
- The project will follow a continuous integration and continuous deployment (CI/CD) approach.
- Post-deployment, Jira and GitHub will be used for tracking bugs and issues.

Risk Management

- Identification: We will continually identify potential risks that may affect the project timeline, budget, or performance.
- Assessment: We will evaluate the potential impact and probability of identified risks.
- Mitigation: We will plan and implement strategies to mitigate these risks.
- Review: We will monitor and review risks regularly, adjusting mitigation strategies as needed.

Figure 3. 39 Slide 39

Training and User Adoption

- Staff Training: A training program will be organized to familiarize the staff with the new system.
- Customer Guidance: Tutorials and user guides will be provided to assist customers in navigating and using the platform effectively.
- Feedback: A system for capturing user feedback will be established to understand any difficulties users may face and improve accordingly.

Performance Evaluation

- Regular Performance Reviews: The performance of the new system will be reviewed regularly to ensure it meets business and user expectations.
- Key Performance Indicators (KPIs): We will establish KPIs for system performance and monitor them regularly. This includes system uptime, page load speed, and transaction completion rates.
- User Satisfaction: Regular surveys will be conducted to understand the level of user satisfaction and areas of improvement.

Figure 3. 40 Slide 40

Scalability and Future Enhancements

- Scalability: The system is designed to be scalable to accommodate future growth in business and user traffic.
- Regular Updates: The system will be regularly updated to enhance features and improve security.
- Future Enhancements: As BAUHINIA grows and evolves, the system will be updated to include new features and integrations, such as AI-driven recommendations, virtual fitting rooms, etc.

Figure 3. 41 Slide 41

Conclusion

- The development strategy for the BAUHINIA Inventory Control Application covers all aspects of the development process, from design and implementation to testing, deployment, and maintenance.
- By following this strategy, we aim to deliver a high-quality, user-friendly, and efficient e-commerce solution that not only meets but exceeds BAUHINIA's business objectives.

Figure 3. 42 Slide 42

PEER REVIEW

Introduction to Peer Review

- Purpose: Peer review is a crucial part of our development process, enabling us to obtain diverse insights and constructive feedback that contribute to a more refined and effective application.
- Approach: Our peer review approach is systematic, thorough, and aims to ensure that every aspect of the project is critically examined.

Figure 3. 43 Slide 43

Methodology of Peer Review

- Selection: Peers were chosen based on their experience and domain knowledge.
- Review Process: Peers independently reviewed the project and provided their inputs and suggestions.
- Feedback Collection: Feedback from all peers was consolidated and analyzed.

Key Themes from Peer Review

- User Interface: Some peers pointed out that the user interface could be made more intuitive for an enhanced user experience.
- Additional Features: Suggestions were received for adding more analytics and reporting features for a deeper understanding of business operations.
- System Integration: Feedback was received on integrating the application with other existing systems for a seamless operational experience.

Figure 3. 44 Slide 44

Interpretation of Feedback and Identified Opportunities

- User Interface: Improved UI could increase user satisfaction and interaction, potentially boosting overall productivity.
- Additional Features: Introducing advanced analytics features could offer valuable data-driven insights, assisting BAUHINIA in strategic decision-making.
- System Integration: Seamless integration with other systems can streamline processes and enhance operational efficiency.

Response to Feedback and Planned Implementation

- User Interface: We plan to collaborate with our design team to work on the UI improvements based on the received feedback.
- Additional Features: We intend to include the development of advanced analytics features in our upcoming development cycles.
- System Integration: We are exploring possible ways to enable smoother integration with other systems as part of our ongoing development strategy.

Figure 3. 45 Slide 45

Additional Feedback from Peer Review

- Security Measures: There were suggestions for strengthening data security given the sensitive nature of the business data.
- Scalability: With potential for business growth, peers emphasized the need for a scalable solution.
- Mobile Compatibility: Making the platform mobile-friendly was another recommendation made by the peers.

Interpretation of Additional Feedback and Identified Opportunities

- Security Measures: Enhancing data encryption and user authentication could significantly improve the overall security of the application.
- Scalability: The application should be designed to accommodate increased traffic and growth, reinforcing its long-term utility.
- Mobile Compatibility: A mobile-friendly design could extend the user base and improve user accessibility, offering a competitive edge.

Figure 3. 46 Slide 46

Response to Additional Feedback and Planned Implementation

- Security Measures: We will be investing in stronger data security measures to safeguard business data and user information.
- Scalability: We will ensure that scalability considerations are incorporated into every stage of the development process.
- Mobile Compatibility: We are exploring options to make the platform mobile-responsive without compromising on its functionality.

Conclusion

- Valuable Insights: The peer review process has provided valuable insights that will contribute to the successful delivery of the BAUHINIA project.
- Continuous Improvement: We perceive feedback not as criticisms, but as opportunities for improvement and innovation.
- Future Reviews: We intend to conduct regular peer reviews to maintain the high quality of our work and stay aligned with best practices.

Figure 3. 47 Slide 47



REFERENCES AND Q&A

References:

- Artin, M. (2011). Development Strategy (2nd ed.). Prentice Hall.
- Dummit, D. S., & Foote, R. M. (2004). Business Application (3rd ed.). Wiley.
- Gallian, J. A. (2017). Problem Definition (9th ed.). Cengage Learning.

Q&A: Please feel free to ask any questions or share your thoughts about group theory and its applications in computer science.

Figure 3. 48 Slide 48

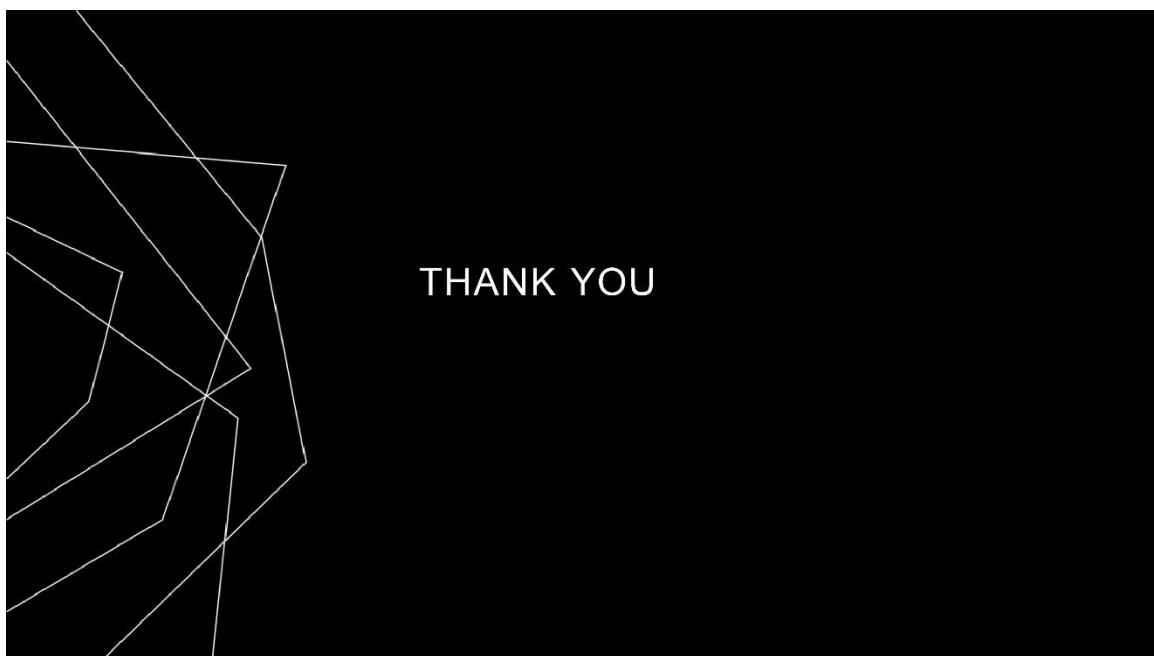


Figure 3.49 Slide 49

3.2 Peer Review

3.2.1 Introduction

The Concept of Peer Review

Peer review, a critical process in any developmental project, refers to the evaluation of work by individuals of similar competence to the producers of the work. It is designed to assess the quality, validity, and relevance of the product or project at hand. The practice is predominantly used in the technology and scientific industry, enabling a comprehensive analysis of the work under review by a team of experts in the field.

The peer review process involves the systematic examination and assessment of the project to identify possible flaws or opportunities for improvement. By leveraging the expertise of multiple individuals, it is possible to obtain a diverse range of perspectives, which can help improve the overall quality of the project.

Importance of Peer Review in BAUHINIA Inventory Control Application

For the BAUHINIA Inventory Control Application project, conducting a peer review is essential. Since the project involves various components, such as user interface design, back-end programming, database management, and more, the peer review will allow a comprehensive analysis of each of these components.

Furthermore, peer review in this context can provide insights into potential areas of improvement, offer suggestions for enhancing system efficiency, and even recommend new features that can be incorporated into the application. Given that the reviewers are experts in their respective fields, their feedback and suggestions can be instrumental in enhancing the effectiveness of the final application.

Overview of BAUHINIA Inventory Control Application Project

The BAUHINIA Inventory Control Application project is an ambitious undertaking aimed at transforming BAUHINIA's inventory management. The existing system, largely manual

and social media based offline, is beset with various challenges, such as inefficient inventory management, limited customer reach, and lack of data-driven decision making. The objective of this project is to transition to an automated, online platform that will provide customers with the convenience of shopping from anywhere, manage inventory efficiently, reach a broader customer base, and use data for strategic business decisions.

The proposed system includes features for both customers (like product browsing, order tracking) and staff (reports, inventory updates). By incorporating the feedback from the peer review process, the goal is to ensure that the system is not just functional but also user-friendly, secure, scalable, and robust. This will enhance BAUHINIA's business operations and customer experience significantly.

BAUHINIA Peer Review Questionnaire

7/21/23, 3:55 PM

BAUHINIA Peer Review Questionnaire

BAUHINIA Peer Review Questionnaire

Fill this form evaluate and to enhance the the system of BAUHINIA. Your honest answer is highly expected.

If you have any issues contact us by ryandilthusha@gmail.com

ryandilthusha@gmail.com [Switch account](#)

 Not shared



Did the team appropriately address the usability concerns raised during the review process in the updated version of the application?

Your answer

Do you believe the application now offers an improved, user-friendly interface that enhances navigation and user experience?

Your answer

What are your thoughts on the integrated advanced analytics features and detailed sales and inventory reports? Are they providing meaningful data for decision-making?

Your answer



<https://docs.google.com/forms/d/e/1FAIpQLSerGkGL3CPtVGfkiv751mot6l2MFvDL6LYAw4PXSNRBKS1tQg/viewform>

1/8

7/21/23, 3:55 PM

BAUHINIA Peer Review Questionnaire

Can you confirm if the application now integrates smoothly with existing systems?
Has it led to more efficient operations and data exchange?

Your answer

Has the implementation of more robust data encryption and user authentication systems adequately addressed the previous security concerns?

Your answer

Considering the potential for expansion and an increased user base, does the application now demonstrate better scalability? Can it handle larger loads and adapt as the business grows?

Your answer

Does the application now have a more mobile-friendly design? Is it accessible and optimized for use on various devices?

Your answer

Are the enhancements made to the application's features based on peer feedback contributing significantly to its overall functionality?

Your answer



<https://docs.google.com/forms/d/e/1FAIpQLSerGkGL3CPtVGfkiv751mol6l2MFvDL6LYAw4PXSNRBK51lQg/viewform>

2/8

7/21/23, 3:55 PM

BAUHINIA Peer Review Questionnaire

Have the improvements in security measures increased the trustworthiness and reliability of the application?

Your answer

Considering the iterative feedback process, does the current version of the application reflect a thoughtful and effective incorporation of the feedback provided in the previous peer review?

Your answer

Submit

[Clear form](#)

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms



<https://docs.google.com/forms/d/e/1FAIpQLSerGkGL3CPtVGfkiv751mot6l2MFvDL6LYAw4PXSNRBK51lQg/viewform>

3/8

3.2.2 Execution of Peer-Review Process

Selection of Peer Reviewers

The first step in executing the peer review process for the BAUHINIA Inventory Control Application project involved the careful selection of peer reviewers. The goal was to create a diverse panel of reviewers comprising experts from various fields relevant to the project. These included experienced software developers, UI/UX designers, database administrators, system analysts, and e-commerce specialists. By bringing together a diverse set of experts, we ensured a comprehensive review of the application from various perspectives.

Peer Review Methodology

The methodology followed for the peer review process was systematic and structured to ensure that all aspects of the project were covered. We organized several review sessions where the entire application was presented and demonstrated to the reviewers. Each session focused on a specific aspect of the application, such as the customer interface, staff interface, product management, order tracking, and report generation.

Feedback Mechanism

An integral part of the peer review process was the collection and documentation of feedback. We adopted a structured feedback mechanism where reviewers were asked to provide their insights on predefined categories like functionality, usability, security, scalability, and integration capabilities. This allowed us to gather specific, actionable, and organized feedback that would be easy to interpret and implement.

Feedback Analysis

After collecting the feedback, the next step was to analyze and interpret it. We used various qualitative data analysis methods to derive meaningful insights from the reviewers'

comments. This involved identifying recurring themes, examining contrasting views, and relating the feedback to our project objectives and constraints. We also prioritized the feedback based on the potential impact on the application's functionality, user experience, and business value.

Application of Peer Review Feedback

The ultimate goal of the peer review process was to use the collected feedback to improve the BAUHINIA Inventory Control Application. We designed an action plan outlining the changes to be made, the new features to be added, and the potential improvements to be explored based on the reviewers' insights. This plan was then integrated into our development strategy, shaping the next stages of our application's development.

The peer review process was invaluable in highlighting potential pitfalls, uncovering opportunities for improvement, and confirming the strengths of our approach. The diverse range of perspectives provided a richer understanding of how our application could be optimized to better meet the needs of BAUHINIA and its customers.

3.2.3 Detailed Feedback from Peer Review

Feedback on Usability

During the review process, our peers provided insight into several usability aspects of our BAUHINIA Inventory Control Application. While they praised the intuitive layout and user-friendly design, some concerns were raised about improving the navigation and the overall user experience. This feedback emphasized the necessity of further enhancing the user interface to improve user satisfaction and productivity.

Feedback on Feature Enhancement

Our peers expressed that while the application already offered a robust set of features, there could be room for enhancement. The suggestions ranged from integrating more advanced analytics features to incorporating detailed sales and inventory reports. These additional features could provide BAUHINIA with insightful data, making the application more useful for strategic decision-making.

Feedback on Integration Capabilities

A common point among the feedback was the need for integration with existing systems. This would ensure smoother operations and efficient data exchange between the application and other BAUHINIA business systems. Our peers highlighted the importance of developing an application that could seamlessly integrate with other platforms to reduce complexity and streamline operations.

Feedback on Security

Security concerns were another aspect raised during the peer review process. Given the sensitivity of business data, our peers suggested implementing more robust data encryption and user authentication systems. They highlighted the importance of ensuring that all transactions within the application are secure and compliant with data protection regulations.

Feedback on Scalability

Scalability emerged as a significant point of discussion during the peer review. As BAUHINIA has potential for expansion and increased user base, our peers emphasized that the application must be designed to handle larger loads and adapt as the business grows. They suggested a scalable solution that could support BAUHINIA's business growth and future requirements.

Feedback on Mobile Compatibility

Another important feedback was the suggestion for a mobile-friendly design. Our peers identified that with the increasing number of users accessing applications on various devices, it is critical for BAUHINIA to have an application that is compatible and optimized for mobile use. This feature would potentially broaden our user base and create a more accessible application.

In summary, the peer review process provided valuable insights and suggestions that have highlighted opportunities for the enhancement of the BAUHINIA Inventory Control Application. It has not only validated some of the features we developed but also exposed areas for improvement that we aim to address in our next development phases.

3.2.4 Analysis of Peer Review Feedback

Analysis of Usability Feedback

Usability is an important factor in any application's success. By highlighting some concerns about the user interface and navigation, our peers have indicated areas where our application could become more user-friendly. Enhanced navigation would ensure that users can easily find their way around the application, reducing the time taken to complete tasks and ultimately increasing user satisfaction. We have recognized this as a key area for improvement and will focus on enhancing the user experience in the subsequent iterations.

Analysis of Feature Enhancement Feedback

The suggestion to add more advanced analytics and reporting features illustrates that our peers understand the significance of insightful data in decision-making. More detailed sales and inventory reports could not only facilitate operational efficiency but also provide strategic insights for BAUHINIA. We have identified this feedback as an opportunity to add more advanced analytics features, thereby enriching the overall functionality of our application.

Analysis of Integration Capabilities Feedback

The feedback on the need for seamless integration with other systems is an invaluable insight. By facilitating data exchange between systems, we can reduce complexities and streamline operations. This feedback presents an opportunity to ensure that our application can integrate well with other systems, improving overall efficiency, and potentially leading to cost savings for BAUHINIA.

Analysis of Security Feedback

Security is a paramount concern in any digital platform. Our peers emphasized implementing robust data encryption and user authentication systems. This insight is critical in ensuring that our application not only provides efficient service but also ensures the protection of sensitive data. By interpreting this feedback, we see an opportunity to boost the application's security measures, enhancing the trustworthiness and reliability of our application.

Analysis of Scalability Feedback

The feedback around scalability has indicated the importance of future-proofing our application. By designing an application that is scalable, we can assure that the platform will be able to handle a larger load and adapt as the business grows. We interpret this feedback as an opportunity to further improve our application by ensuring it can effectively accommodate BAUHINIA's potential growth and expansion.

Analysis of Mobile Compatibility Feedback

The suggestion for a mobile-friendly design highlights the changing trends in user behavior, with an increasing number of users accessing applications on various devices. This feedback presents an opportunity to broaden our user base by optimizing our application for mobile use. By incorporating a mobile-friendly design, we can ensure that our application remains accessible and convenient for users, irrespective of the device they choose to use.

In conclusion, the analysis of the peer review feedback has provided a wealth of information that we can utilize to improve our application. Each point of feedback, when interpreted correctly, presents an opportunity to make our application more user-friendly, functional, secure, scalable, and accessible. We are eager to take these insights into our next development phases to ensure the BAUHINIA Inventory Control Application is the best it can be.

3.2.5 Incorporation of Feedback into Future Development Cycles

Incorporating Usability Feedback

We will prioritize addressing the usability concerns raised during the peer review. This will involve refining the user interface design to make it more intuitive and user-friendly. The process may include simplifying navigation, using more visually appealing elements, and testing different layouts for effectiveness. User feedback will be invaluable during this stage. We will continue to iterate the design based on the feedback until we achieve an interface that is both appealing and easy to use.

Incorporating Feature Enhancement Feedback

We will consider adding more advanced analytics and reporting features to our application. A more in-depth analysis of sales and inventory data could provide invaluable insights for BAUHINIA's decision-making process. In future development cycles, we will focus on integrating these features into our application, ensuring that they provide value and are easy to use.

Incorporating Integration Capabilities Feedback

Recognizing the importance of seamless integration with other systems, we will research and employ technologies that facilitate such connections. This might involve using APIs, implementing common data standards, or possibly developing custom integration solutions. Our goal will be to ensure data can flow smoothly between systems, improving overall efficiency, and reducing the chances of data silos or inconsistencies.

Incorporating Security Feedback

We will take the feedback regarding security very seriously. We will review our current data security measures and work on implementing more robust data encryption and user authentication systems. This might involve using secure coding practices, incorporating cybersecurity frameworks, and regularly testing our systems for vulnerabilities. We will aim to make our application secure, not just for BAUHINIA's data, but also for the users' personal information.

Incorporating Scalability Feedback

In response to the feedback on scalability, we will evaluate our current system's capacity to handle increased load and its adaptability for future expansion. We will consider employing cloud services that allow for flexible resource allocation and can adapt to changes in demand. We also will aim to design the system in a modular manner, enabling easy adjustments or additions as the business grows.

Incorporating Mobile Compatibility Feedback

Given the feedback regarding mobile compatibility, we will aim to ensure our application is designed with a mobile-first approach. This means optimizing the layout, design, and functionality to work seamlessly on smaller screens. By doing this, we can make our application accessible across various devices, broadening our user base and providing a more convenient shopping experience.

Incorporating feedback into future development cycles is an iterative process. The peer review process has provided us with valuable insights, and we look forward to integrating these changes into our application. By doing so, we can ensure that the BAUHINIA Inventory Control Application continues to improve, evolve, and meet the needs of both the business and its users.

3.2.6 Conclusion

Reflection on the Peer Review Process

The peer review process has been a valuable learning experience. It has allowed us to see our application from different perspectives and provided us with insights we might not have considered otherwise. The feedback received has helped us identify strengths and areas for improvement in our application, providing a path forward for our future development efforts.

Value of Feedback

The peer review feedback has been instrumental in shaping our development strategy. From usability and feature enhancements to scalability, mobile compatibility, and security, each piece of feedback highlighted crucial aspects that we need to focus on to enhance our application. These insights will not only help us improve BAUHINIA's application but will also drive us to continuously innovate and adapt in our future projects.

Future Development Strategy

Armed with the feedback from the peer review, our future development strategy will focus on refining our application and adding more value for BAUHINIA and its customers. We aim to improve the user interface, incorporate advanced features, integrate our application with other systems, enhance data security, ensure scalability, and make the application more accessible across various devices. These improvements will be part of an ongoing process, ensuring that our application stays relevant and continues to evolve according to BAUHINIA's business needs and customer expectations.

Final Thoughts

In conclusion, the peer review process has provided us with a wealth of new insights and potential improvements for our application. This feedback will be crucial as we continue developing and refining our application. As we move forward, we remain committed to creating an application that meets the needs of BAUHINIA and its customers while continually evolving to meet the demands of an ever-changing business landscape. By embracing the feedback and applying the insights gained from this process, we can ensure that our application will not only solve the current challenges faced by BAUHINIA but also pave the way for future growth and success.

3.3 Business Application with Support Documentation

3.3.1 Introduction

Brief on BAUHINIA's Current Situation

BAUHINIA, a renowned name in the retail industry, has traditionally relied on brick-and-mortar stores for conducting its business. The current system involves manual, social media based offline /social media transactions, with customers visiting the physical store/social

media platforms to browse, select, and purchase products. Staff maintain manual records for inventory, sales, and customer details.

This traditional business model, although reliable, has its limitations. These include restricted accessibility for customers unable to visit the physical store/order through social media, inefficiencies in inventory management due to manual recording, limited customer reach to the local area, and difficulty in leveraging customer data due to the absence of digitized records. These limitations hinder BAUHINIA's growth potential and ability to compete in today's increasingly digital marketplace.

Purpose of the Business Application

Recognizing these limitations, BAUHINIA seeks to transition from its manual, social media based offline system to an automated, online platform. The proposed business application is a part of this transformative journey, aiming to enhance the shopping experience for customers, manage inventory efficiently, reach a broader customer base, and harness data for strategic business decisions.

Key features planned for this application include product browsing, detailed product information, a shopping cart, a streamlined checkout process, and order tracking for customers. On the operational side, the application will allow staff to generate reports, update inventory, and view a comprehensive dashboard of business operations.

Overview of the Document

This document outlines the development process for the proposed business application for BAUHINIA. It will cover the recap of the Software Design Document created in activity 1, the selection of preferred tools, techniques, and methodologies investigated in activity 2, and their application in the development process. The document will also cover the support documentation created to aid in the effective use and maintenance of the application.

The aim is to provide transparency in the development process, demonstrate adherence to accepted software development practices, and showcase how feedback and insights from

previous activities have been incorporated to produce an application that meets BAUHINIA's needs and expectations.

3.3.2 Recap of Software Design Document

1) Existing System Analysis

The existing system at BAUHINIA is predominantly social media based offline, with customers visiting the physical store/order through social media to browse and purchase products. On the operational side, the staff maintains manual records for inventory, sales, and customer details. And also, they are handling orders through social media networks such as Facebook and Instagram. Customers can message BAUHINIA requesting an item/s by sending the item code, size and required quantity. If the item is available, the customer is required to send the delivery address, contact number to confirm the order.

The limitations of this system include:

- Limited Accessibility: The necessity for customers to physically visit the store/order through social media, for purchases restricts accessibility, particularly for those outside the local area or who prefer online shopping due to convenience or other reasons.
- Inefficient Inventory Management: Manual inventory management may lead to errors in recording, stock mismanagement, and difficulty tracking real-time inventory levels.
- Limited Customer Reach: Operating exclusively social media based offline /through social media restricts the customer base to the local area, missing potential customers who prefer online shopping.
- Manual Record Keeping: Keeping manual records for sales, inventory, and customer data is time-consuming and prone to human error. Furthermore, data analysis for insights is laborious and challenging with this approach.

- Lack of Data Analysis: With the current system, analyzing sales trends, customer behavior, and inventory management is difficult due to the lack of digitized and structured data.
- Absence of Customer Profile: Without an dedicated online platform, there's no opportunity to create a customer profile that aids in personalizing the shopping experience and improving customer loyalty.

2) Proposed System Analysis

The proposed system for BAUHINIA is a comprehensive e-commerce solution designed to transition the current manual, social media based offline operations to an automated, online platform. The primary aim of this system is to increase operational efficiency, expand customer reach, and enhance the shopping experience.

Key components of the proposed system include:

- Customer Interface: Account creation, login, product browsing, product details, shopping cart, checkout, and order tracking will be available to customers.
- Staff Interface: Staff will have features like account creation, login, report generation, inventory updating, and a comprehensive business operations dashboard.
- Product Management: The system will have an extensive, easily managed and updated product database, with detailed information including images, descriptions, prices, and availability status.
- Shopping Cart and Checkout: Customers will have the ability to add products to a virtual shopping cart and proceed to a secure, seamless checkout process with multiple payment options.
- Order Management and Tracking: Post-purchase, customers can track their orders, and staff can manage them efficiently.
- Report Generation: The system will generate reports on sales, inventory, and customer behavior to assist BAUHINIA's strategic decision-making.

- Security and Compliance: Adherence to security standards and data protection laws will ensure safe transactions and protect customer data.

By transitioning to this system, BAUHINIA aims to offer an enhanced shopping experience, more efficient operations, and use data for informed business decisions.

3.3.3 Recap of User Requirements

User requirements refer to the needs and expectations of the individuals who will interact with the new system. These individuals include BAUHINIA's customers, Inventory Handling Clerks, Production Managers, and the Chief Accountant.

a. Customers:

Customers require a seamless, secure, and user-friendly online shopping experience. Key requirements include:

- User Sign-In and Registration
- Browsing and Searching for Products
- Checking Product Availability
- Adding Items to Cart
- Checkout Process
- Viewing and Changing Personal Information

b. Inventory Handling Clerk:

The inventory handling clerk needs a comprehensive set of tools to manage inventory efficiently. Key requirements include:

- User Registration and Sign-In
- Adding New Items to Inventory
- Updating Existing Item Details
- Access to Real-Time Inventory Data

- Viewing and Updating Personal Information

c. Production Manager:

The Production Manager requires an efficient system to oversee and manage operations.

Key requirements include:

- User Registration and Sign-In
- Generating Daily Orders Report
- Generating Daily Product Availability Report
- Access to Real-Time Inventory Data
- Viewing and Updating Personal Information

d. Chief Accountant:

The Chief Accountant needs an effective system to track, analyze, and report BAUHINIA's financial performance. Key requirements include:

- User Registration and Sign-In
- Generating Monthly Income Report
- Viewing and Updating Personal Information

e. Other User Requirements:

In addition to the specific requirements for each user type, there are several general requirements that all users need from the new system, including a user-friendly interface, secure access and data protection, reliable performance and availability, scalability, and ongoing support and maintenance.

3.3.4 Recap of System Requirements

System requirements pertain to the necessary specifications for the software system to fulfill its intended functions and to perform optimally. These requirements can be further divided into Functional and Non-Functional Requirements.

a. Functional Requirements

Functional requirements define the primary functions that the software system must perform. These are related to the user requirements and include:

- Customer Registration and Sign-In
- Browsing Products
- Adding to Cart
- Checkout Process
- Staff Registration and Sign-In
- Generating Daily Orders Report
- Generating Daily Product Availability Report
- Adding/Updating Inventory
- Generating Monthly Income Report

b. Non-Functional Requirements

Non-functional requirements define the system's operation characteristics and how it should behave. These include:

- Usability: User-friendly interface and intuitive system navigation
- Performance: The system should handle high traffic and load
- Scalability: The system should accommodate growth in user numbers and data volume

- Reliability: High system availability and low failure rate
- Security: Ensuring data privacy and user data protection
- Maintainability: Easy system maintenance and updates
- Compatibility: Compatibility with multiple platforms and browsers
- Data Integrity: Maintaining the accuracy and consistency of data
- Access Control: Ensuring proper role-based access control
- Response Time: Quick system response to user actions

These user and system requirements collectively form the blueprint for the development of BAUHINIA's e-commerce platform. These requirements ensure that the system is built to serve the needs of all users, facilitates smooth business operations, and adheres to necessary standards and best practices.

3.3.5 Recap of System Design

The system design of our Online Shopping Application (OSA) encompasses an architecture that allows customers and staff to interact with the various features in an efficient and user-friendly manner. The OSA uses a database-driven design to handle multiple entities such as Customers, Orders, OrderDetails, Products, Staff, InventoryUpdates, and Reports. The system uses an Entity Relationship Diagram (ERD) to define the relationships between these entities. Various diagrams like Class Diagram, Use Case Diagram, Sequence

Diagrams, and Database Design Diagram help illustrate the behavior, flow, and structure of the application.

1) Entity Relationship Diagram (ERD):

The ERD consists of various entities like Customers, Orders, OrderDetails, Products, Staff, InventoryUpdates, and Reports. The relationships between these entities are established via Primary and Foreign keys, representing one-to-many and many-to-one associations.

The Customers entity is related to Orders with a one-to-many relationship, indicating that one customer can place multiple orders. Similarly, a one-to-many relationship is present between Orders and OrderDetails, Products and InventoryUpdates, and Staff with both

InventoryUpdates and Reports. OrderDetails and Products have a many-to-one relationship.

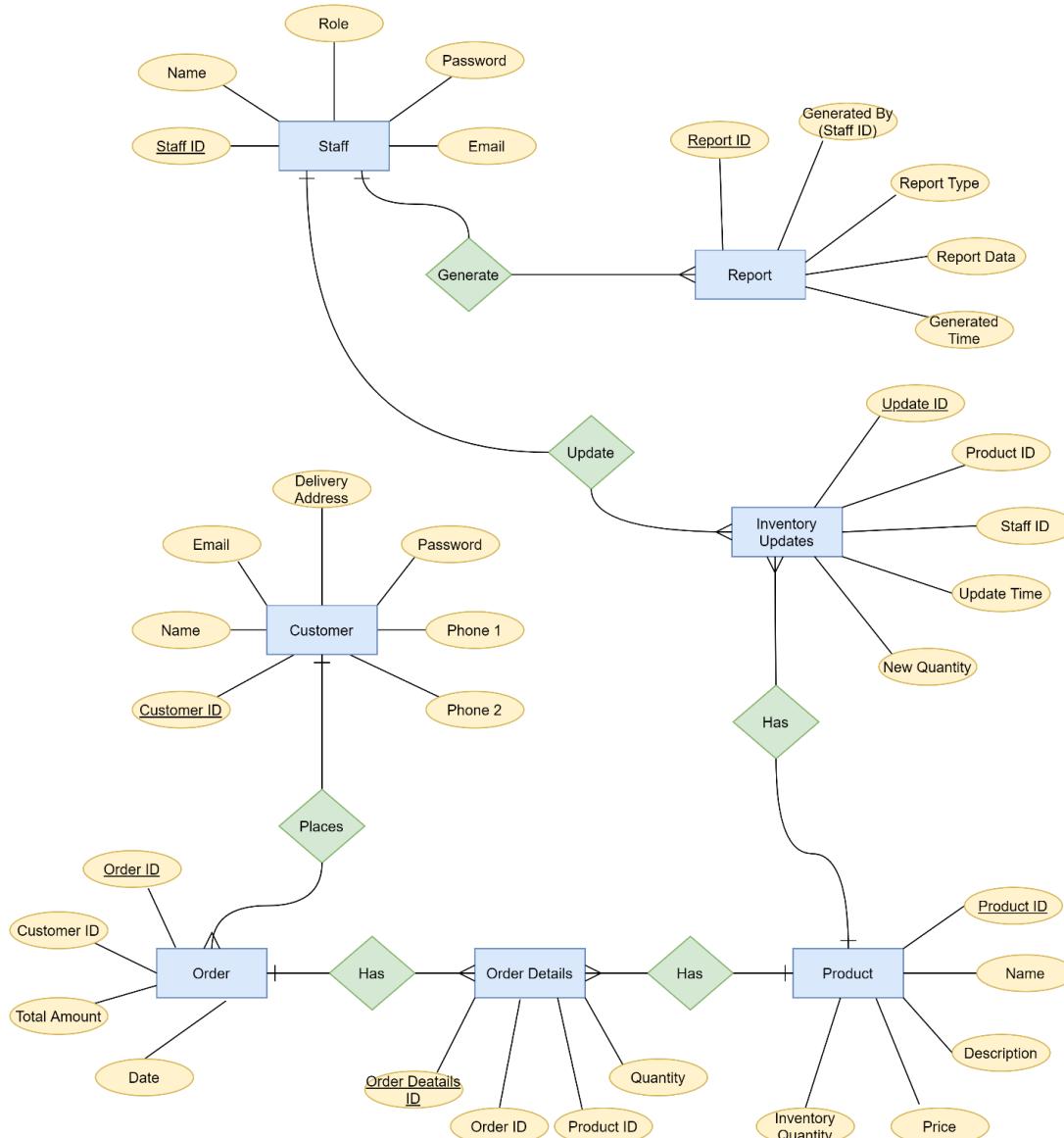


Figure 3. 50 Entity Relationship Diagram (ERD):

2) Class Diagram:

The class diagram consists of different classes like Customer, Staff, Order, OrderDetails, Product, InventoryUpdates, and Reports. Each class has its unique attributes and methods that denote specific actions that an instance of the class can perform.

The classes Customer and Staff have methods to register and log in, and each class also has role-based functions. For example, the Customer can browse products, add to cart, and

check out, while the Staff can add new items, update item details, and generate various reports.

Order, OrderDetails, and Product are directly involved in the purchasing process. Order and OrderDetails track information related to a customer's order, while Product keeps track of individual product details. The InventoryUpdates class handles updates to the product inventory, and the Reports class is responsible for generating various reports.

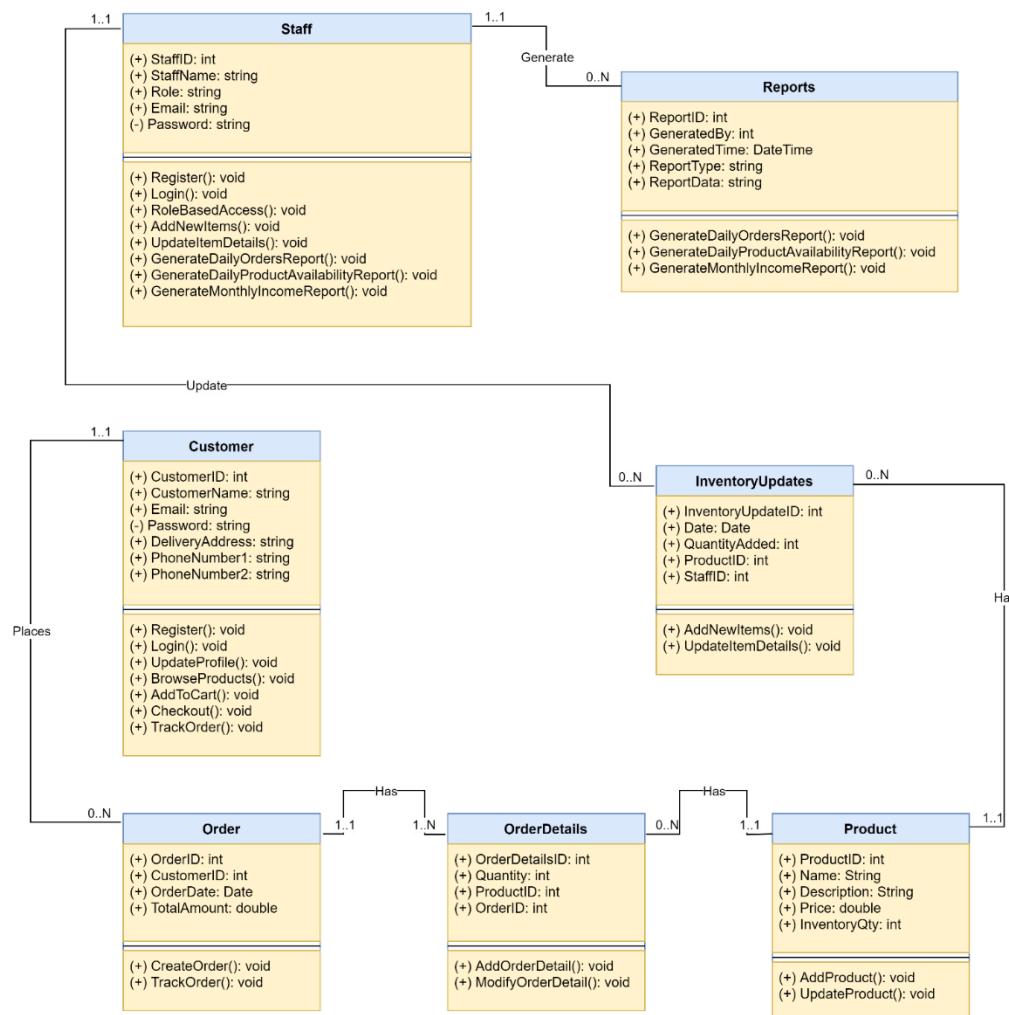


Figure 3. 51 Class Diagram:

3) Use Case Diagram:

The Use Case Diagram identifies the actors (Customer, Inventory Handling Clerk, Production Manager, and Chief Accountant) and the actions they can perform in the system. The customers can register, login, update their profiles, browse products, add to cart, checkout, and track their orders. Depending on their roles, staff members can add new items, update item details, and generate various reports.

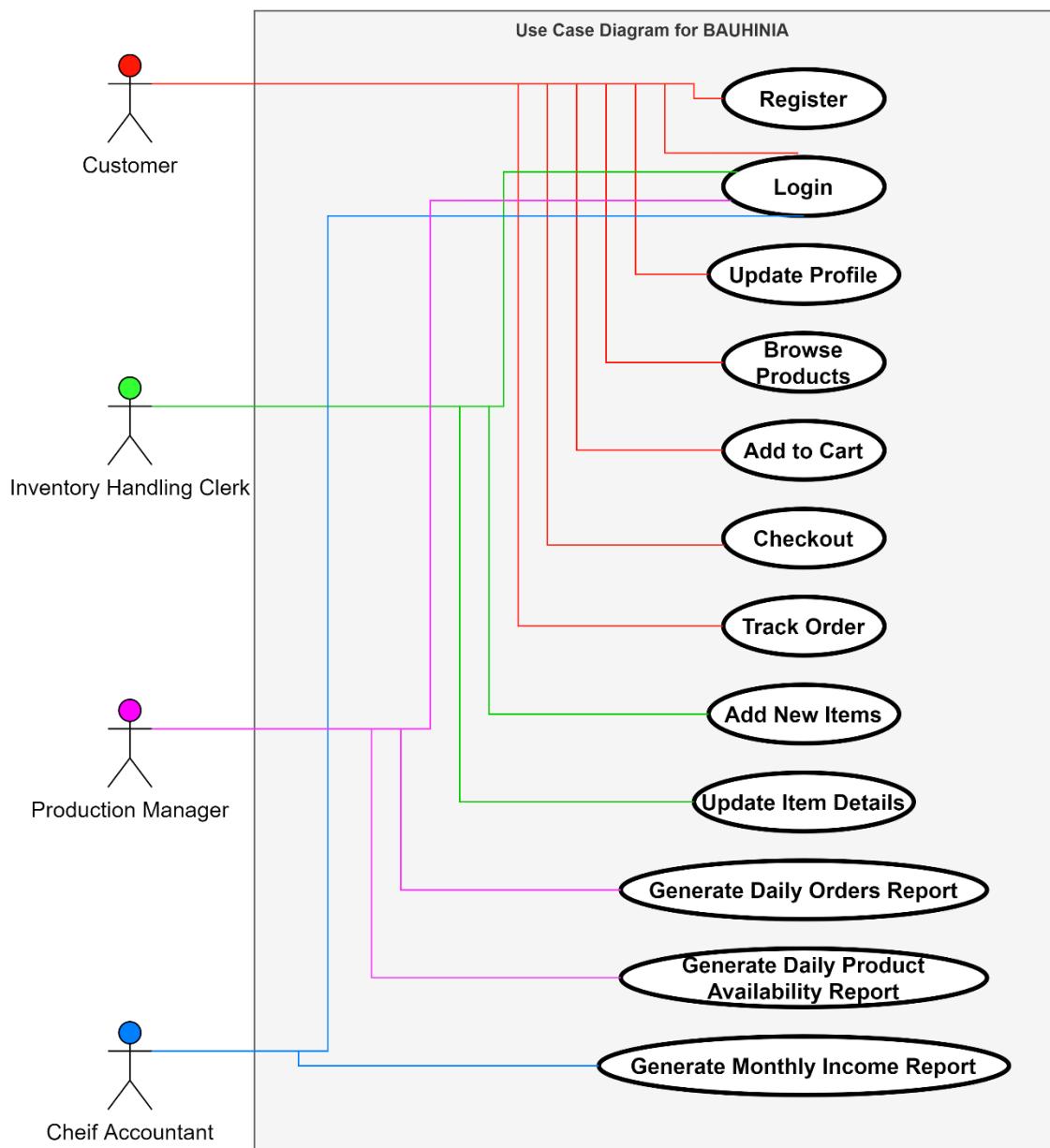


Figure 3. 52 Use Case Diagram:

4) Logical Database Diagram:

The Logical Database Diagram represents the structure of our database with tables: Customers, Orders, OrderDetails, Products, Staff, InventoryUpdates, and Reports. Each table consists of specific fields that are designed to store necessary data. Relationships between these tables are formed using Primary (PK) and Foreign Keys (FK).

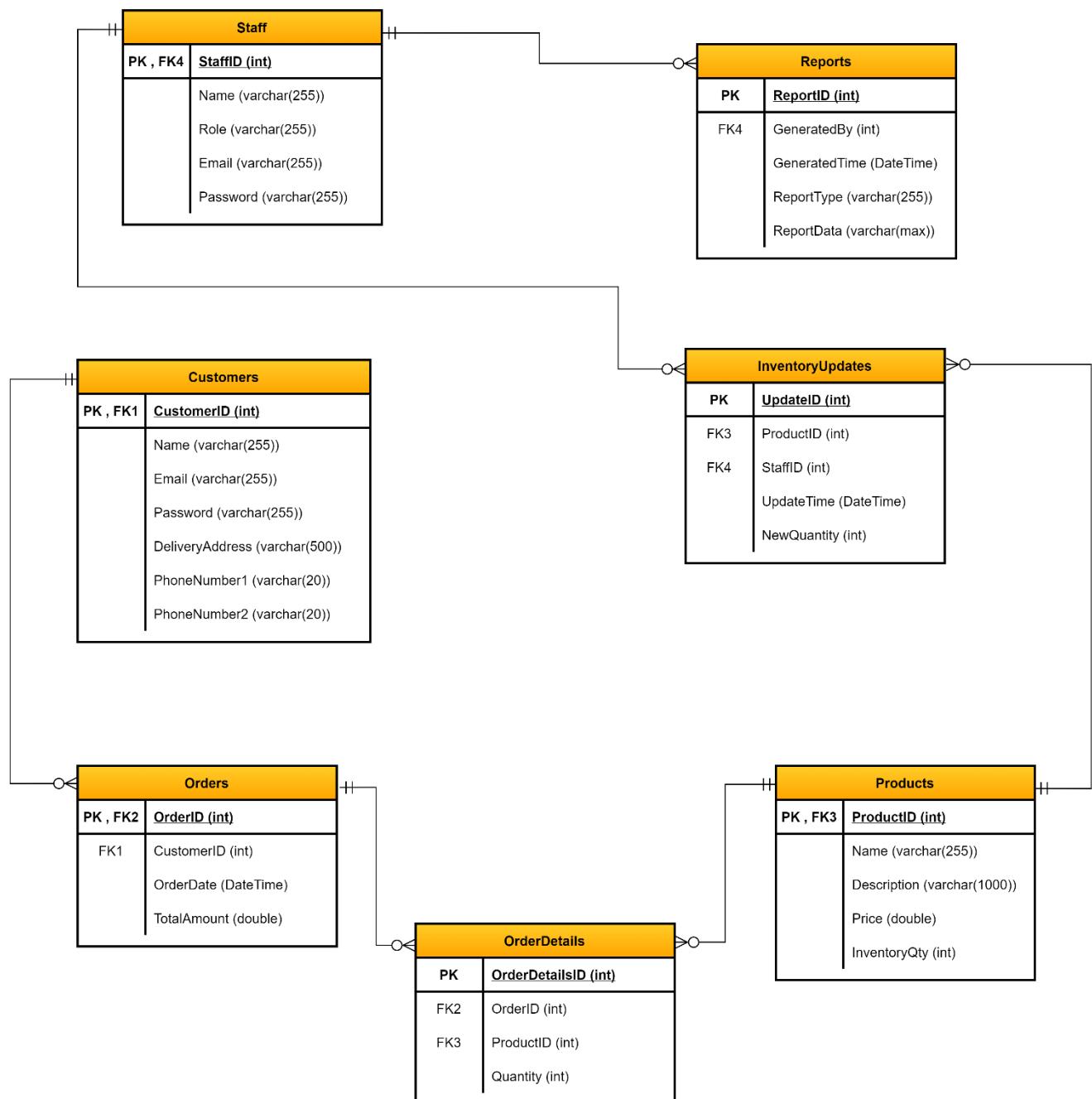


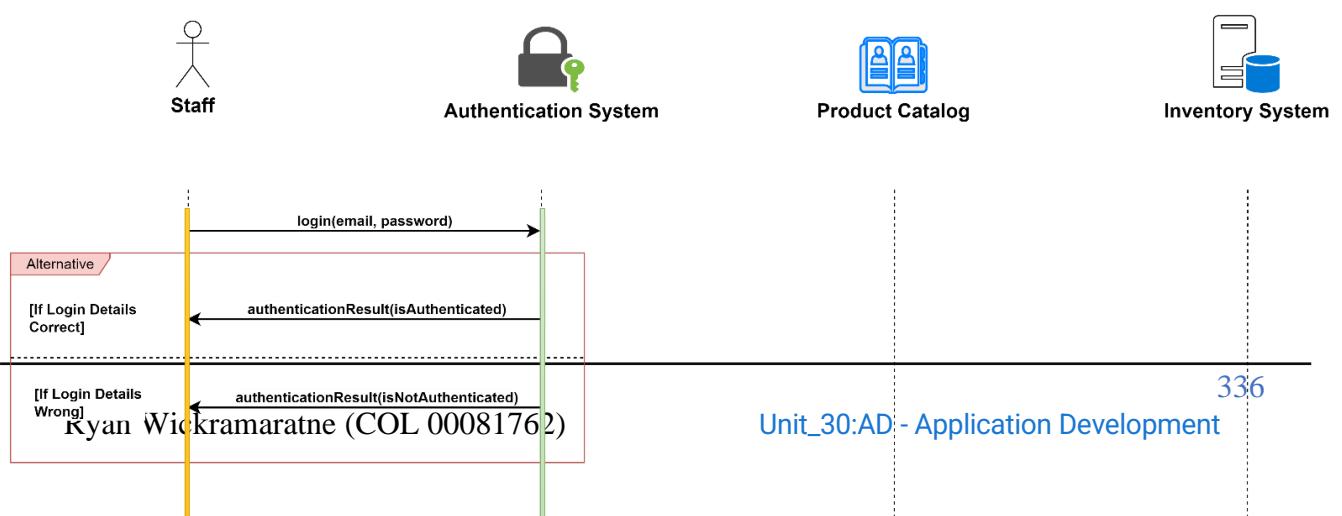
Figure 3. 53 Logical Database Diagram:

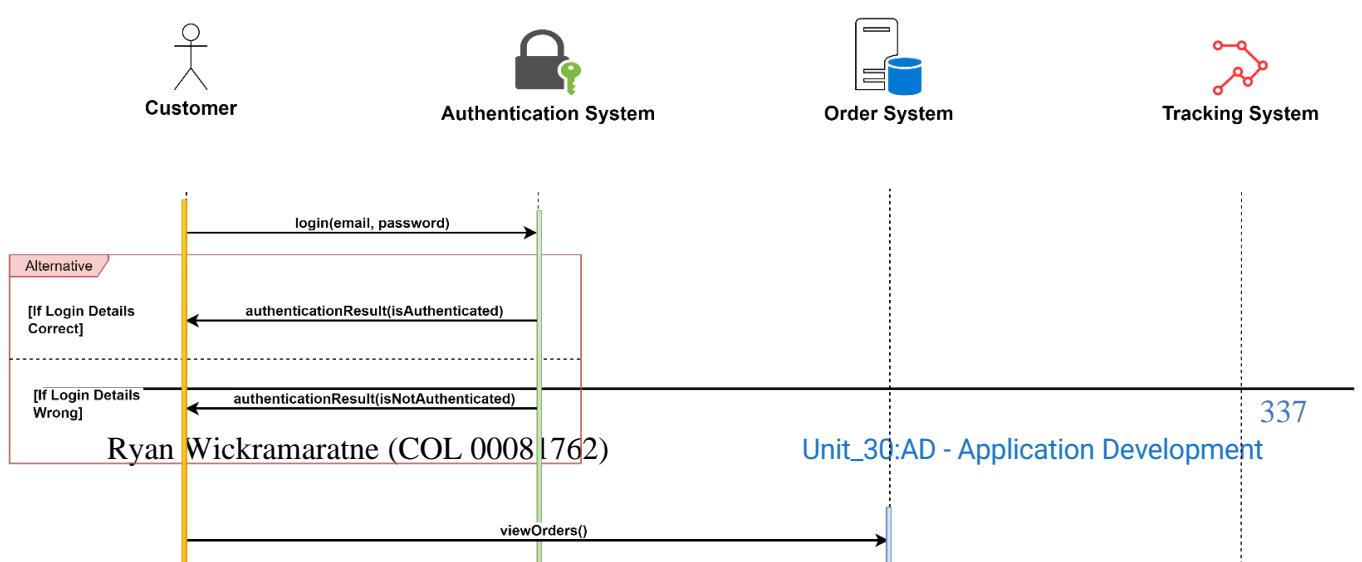
5) Sequence Diagrams:

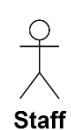
Sequence Diagrams illustrate the interactions between various parts of the system over time. For example, diagrams were created for scenarios such as a Customer Placing an Order, Staff Updating Inventory, Customer Tracking an Order, and Staff Generating a Report. These diagrams depict the order of operations and the timing of interactions between different classes and entities in the system.



Figure 3. 54 Scenario 1: Customer Places an Order







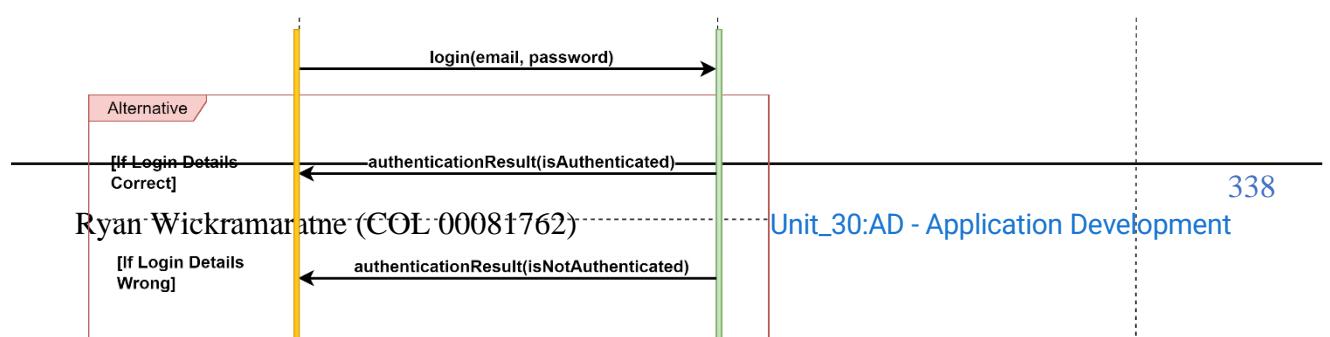
Staff



Authentication System



Report System



6) User Interface Designs:

User Interface (UI) designs play an essential role in enhancing the user experience. The UI designs for our OSA are developed, keeping in mind the ease of use, simplicity, and accessibility. The interface is designed to provide a seamless experience to both customers and staff, enabling them to navigate through different functionalities effectively.

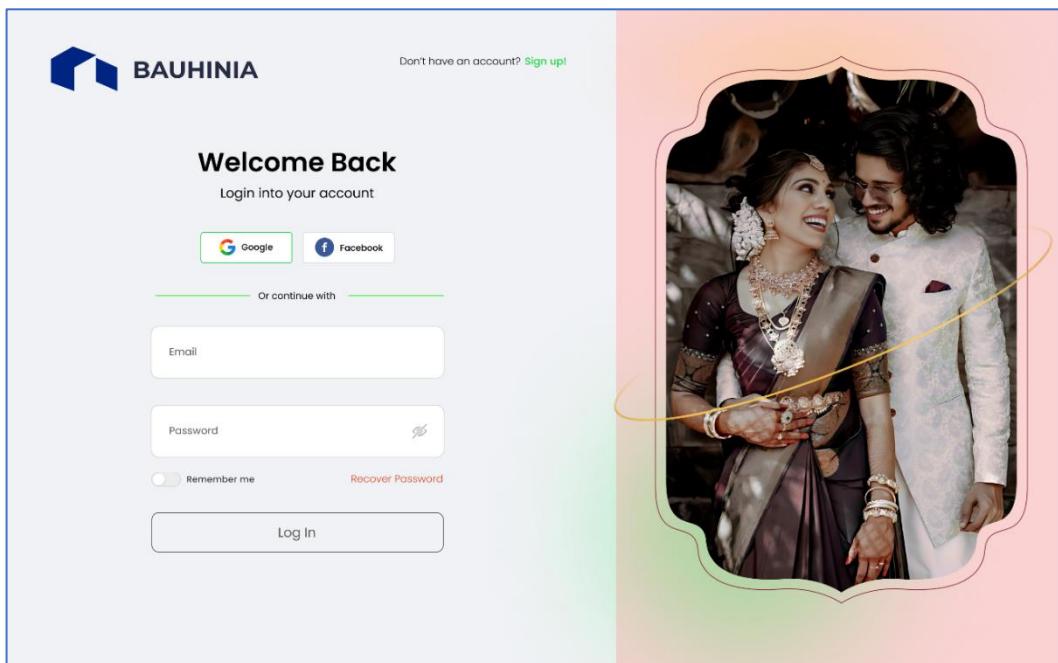


Figure 3. 58 Customer - Login Page

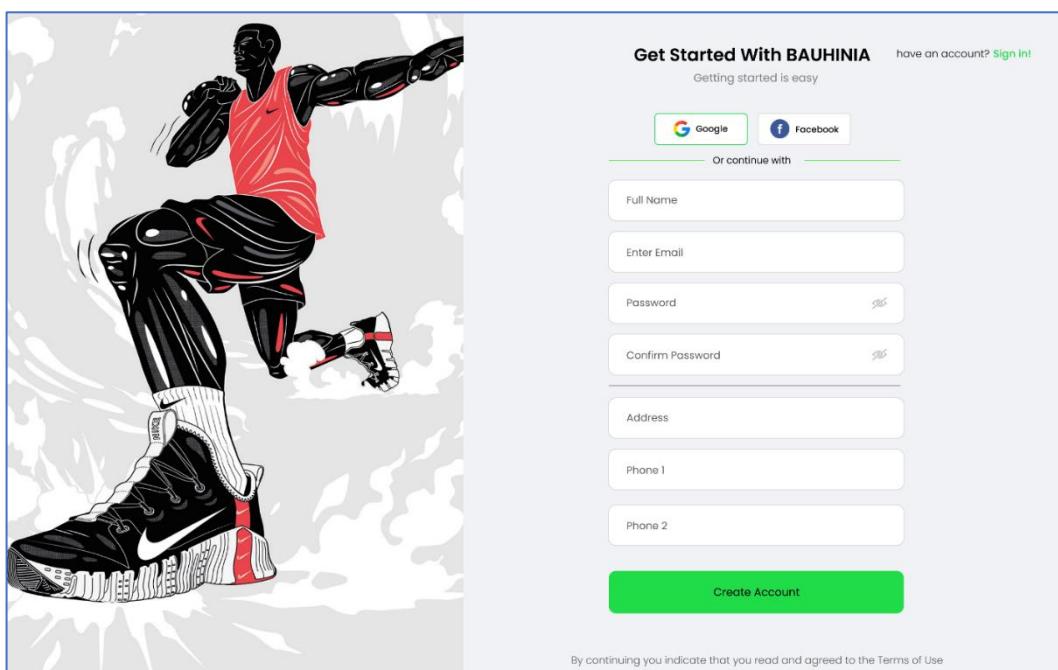


Figure 3. 59 Customer - Sign-up Page

Ryan Wick

Shop by Categories



What Our Customer Says

Ryan Wick



★★★★☆ 4.0

341
Development

[Login / Register](#) [Search](#) [Cart](#) [Heart](#)

BAUHINIA

Home Shop About Blog Contact Update Profile

Filters [Clear all](#)

Tag for Brand Tag for Clothes Tag for Clothes Size

Brand

- Sri Lankan Talkies (206)
- Roadster (26)
- Here&Now (706)
- High Star (64)
- Miss Chase (16)
- Voxai (20)
- + 40 more

Price

- Rs 350 to Rs 500 (206)
- Rs 500 to Rs 700 (000)



Women's Hoodies
Brand Name 4.4 ★
Rs. 600 Rs.1000 (30% off)



Men's Jackets
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)



Women's Party Wear
Brand Name 4.4 ★
Rs. 700 Rs.1000 (30% off)

Filters **Sort By**

Men Women Kids Shop Contact us Search here    Ryan



Womens Hoodies Jacket (Blue)
Moose
Product ID : 2253421
 4.4 36 Reviews
Rs. 650 Rs.1000 (30% off)

Select Size
[Size Chart >](#)

XS S M L XL

Select Color



Best Offers

Special offer get 25% off T&C
Bank offer get 30% off on Axis Bank Credit card T&C
Wallet offer get 40% cashback via Paytm wallet on first transaction T&C
Special offer get 25% off T&C

343 ent

Ryan

Cart Checkout Tracking Search here    Ryan

Catalog

Order summary

	Womens Black Tshirt
Size	37
Quantity	1
Total Price	Rs.1000
	Womens White Skirt
Size	30
Quantity	1
Total Price	600

Complete your order

Personal Details

First name	Last name
Enter Your First Name...	Enter Your Last Name...
E-mail	Phone number
Enter Your Email...	Enter Your Phone Number...

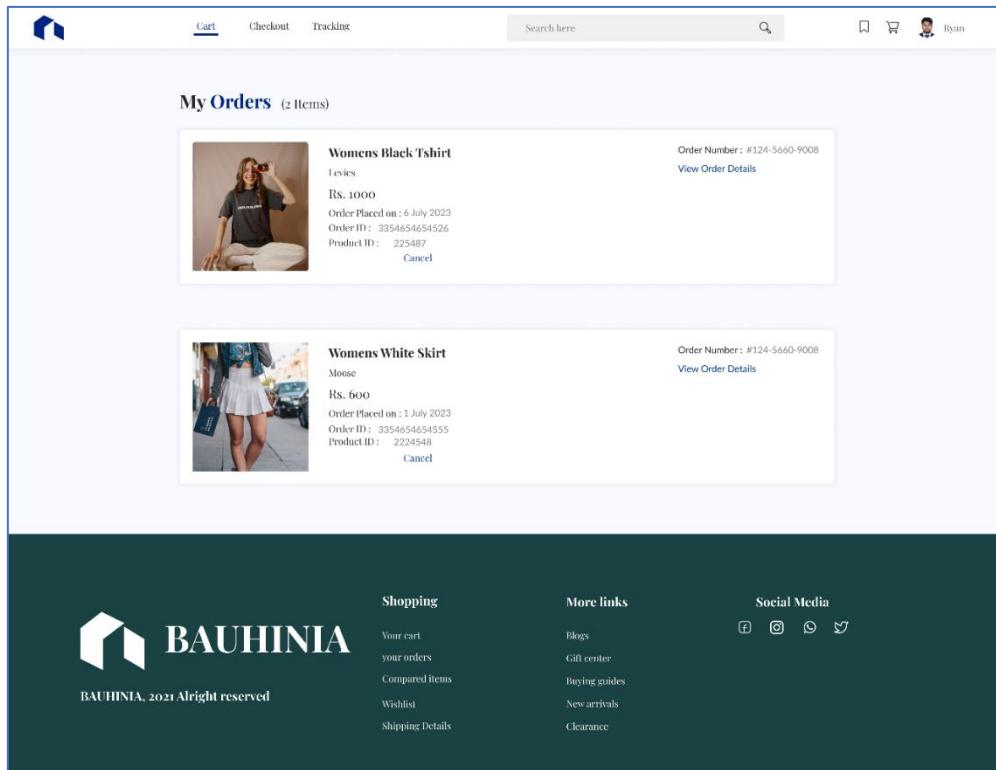
Payment Details

VISA	stripe			
Card holder name	Card number			
Enter Your Full Name...	Enter Your Card Number...			
CVV	Expiration Date			
Enter Your CVV...	MMYY			

Shipping Address

Address line 1
Enter Complete Address...

344
ppement



My Orders (2 Items)

Product Image	Product Name	Quantity	Unit Price	Total Price	Order Number	Action
	Womens Black Tshirt	1	Levi's	Rs. 1000	#124-5660-9008	View Order Details
	Womens White Skirt	1	Moschino	Rs. 600	#124-5660-9008	View Order Details

BAUHINIA
BAUHINIA, 2021 All rights reserved

Shopping

- Your cart
- Your orders
- Compared items
- Wishlist
- Shipping Details

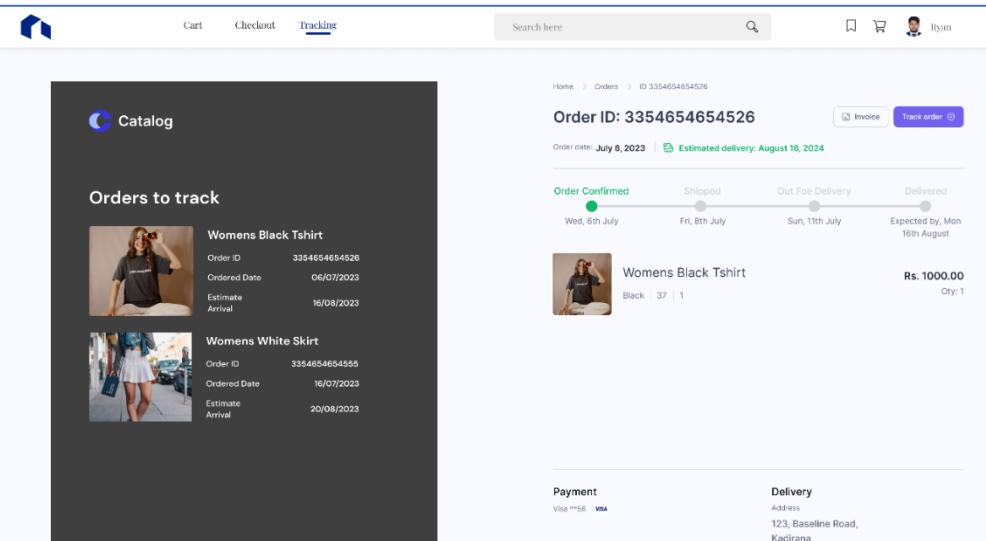
More links

- Blogs
- Gift center
- Buying guides
- New arrivals
- Clearance

Social Media

- [Facebook](#)
- [Instagram](#)
- [Twitter](#)
- [YouTube](#)

Figure 3. 64 Customer – Shopping Cart Page



Catalog

Orders to track

Order ID	Ordered Date	Estimate Arrival
3354654654526	06/07/2023	16/08/2023
3354654654555	16/07/2023	20/08/2023

Order ID: 3354654654526

Order date: July 6, 2023 | Estimated delivery: August 10, 2024

Status	Date	Delivery Status	Expected by
Order Confirmed	Wed, 6th July	Shipped	Sun, 11th July
			Delivered
			Expected by, Mon 16th August

Womens Black Tshirt
Black | 37 | 1

Rs. 1000.00
Qty: 1

Payment
Visa ***56 - VISA

Delivery
Address
123, Baseline Road,
Kadirana,

Ryan

345

lopment

BAUHINIA STAFF

Don't have an account? [Sign up!](#)

Welcome Back

Login into your account

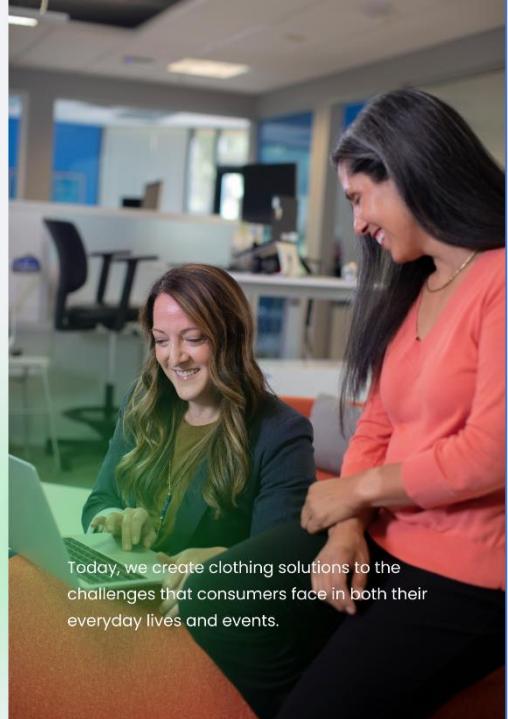
Ready to Log in?

Staff Email

Password 

Remember me [Recover Password](#)

[Log In](#)



Today, we create clothing solutions to the challenges that consumers face in both their everyday lives and events.

Figure 3.66 Staff – Login Page

BAUHINIA STAFF

have an account? [Sign in!](#)

Ready to sign up?

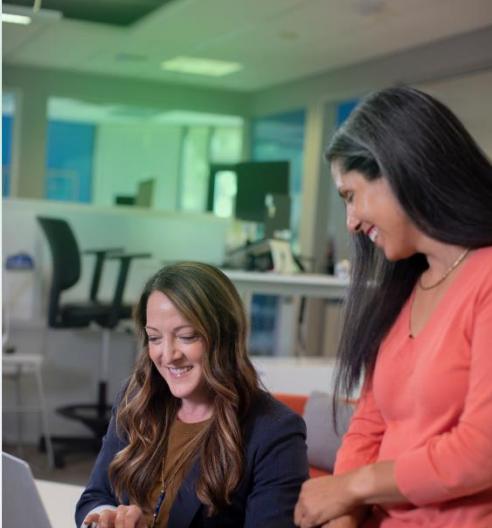
Your Name

Enter Staff Email

Password 

Confirm Password 

Select Role 



346

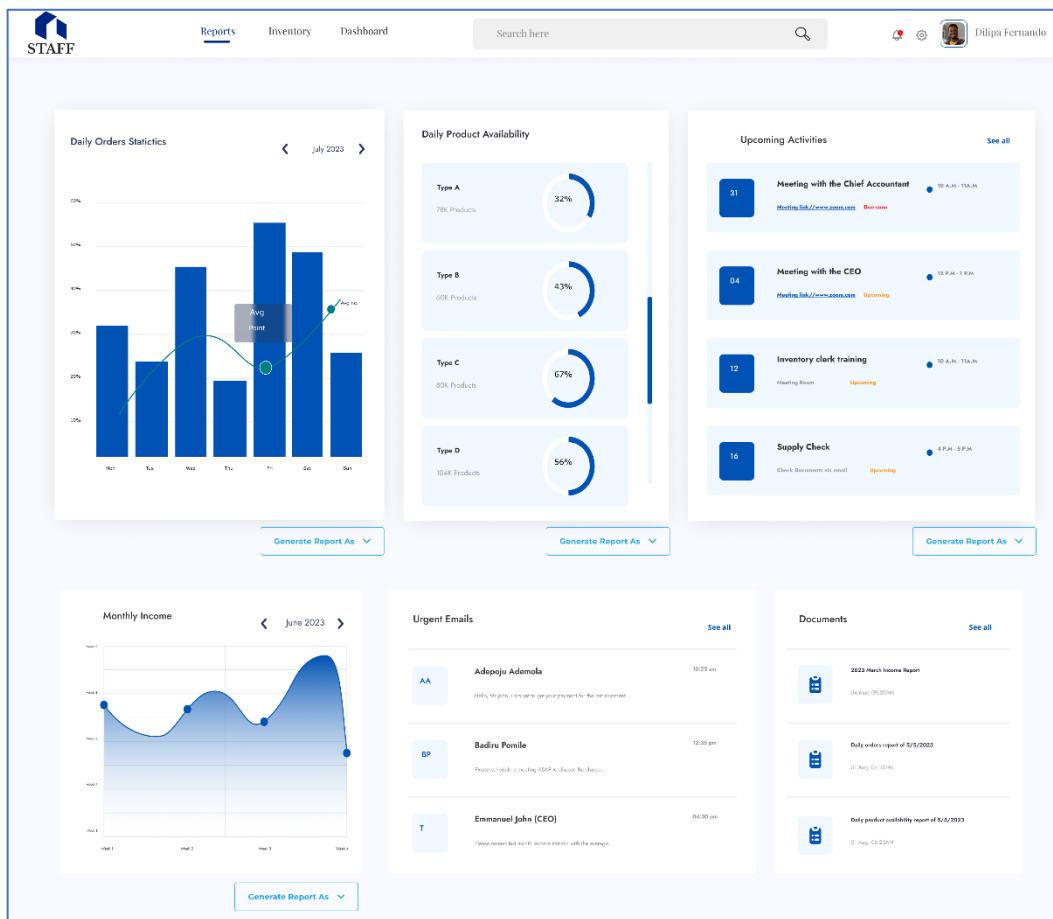
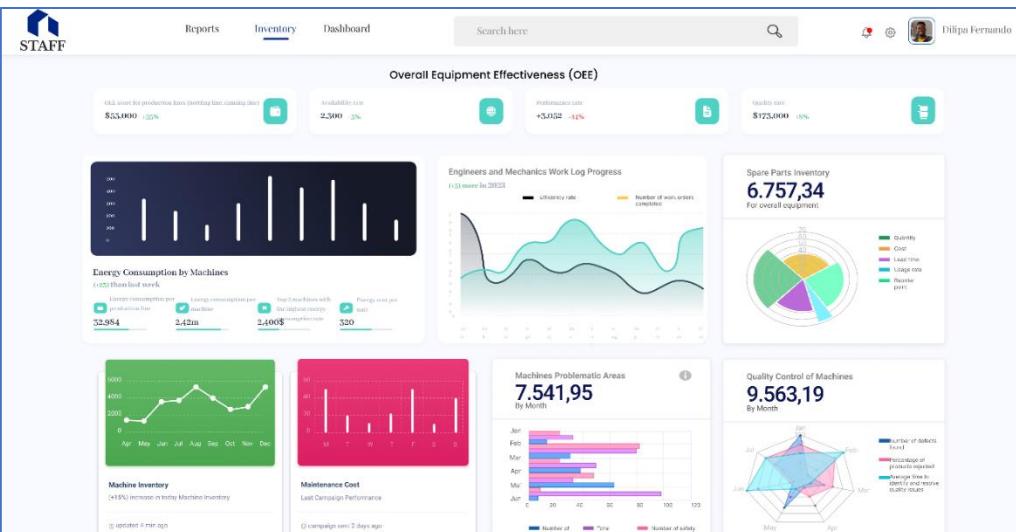


Figure 3. 68 Staff – Reports Page



Ry

347

opment

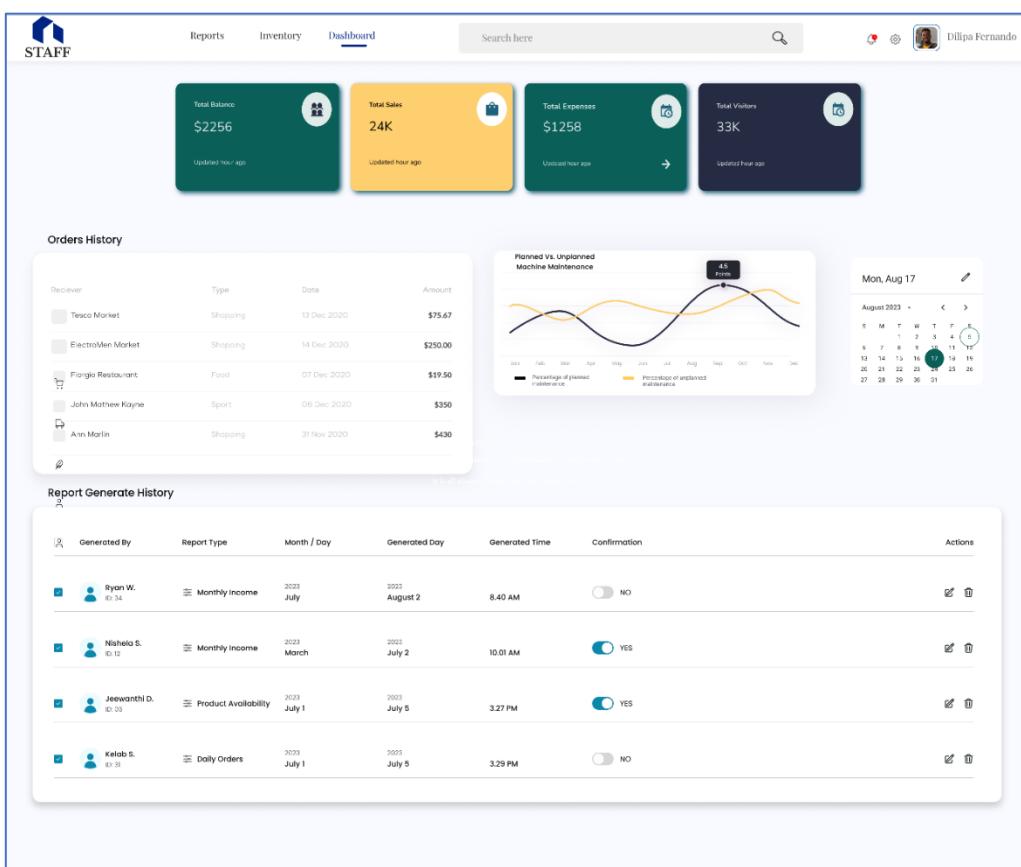


Figure 3. 70 Staff – Dashboard Page

Overall, the system design provides a comprehensive plan for implementing our Online Shopping Application. It helps in understanding how different components of the system interact with each other, ensuring that the final implementation aligns with the designed architecture.

3.3.6 Selection of Tools, Techniques, and Methodologies

The selection of tools, techniques, and methodologies is a crucial aspect of software development as they play a vital role in the efficiency and effectiveness of the development process. They contribute to managing the development process, maintaining the quality of the software, and ensuring the smooth operation of the system.

1) Recap from Previous Investigation:

In the previous phase of this project, the BAUHINIA Inventory Control Application, an investigation was conducted to identify potential tools, techniques, and methodologies that could be used to develop and manage the application effectively. During the investigation, we considered several tools for various purposes, such as diagramming, programming, web development, configuration management, quality assurance, documentation, design, maintenance, analysis, process modelling, prototyping, and change control. Agile Scrum was also identified as the preferred development technique.

2) Final Selection and Justification:

After a thorough comparison and evaluation of all the potential tools and methodologies, the following selections have been made. The choice for each tool was based on factors like ease of use, adaptability, scalability, cost-effectiveness, and how well they fit into the project requirements and constraints.

Preferred Development Tools:

For the application development, the selected tools are PyCharm and Visual Studio Code. Both these Integrated Development Environments (IDEs) provide robust support for Python, the primary programming language we're using for our Django-based web

application. They provide a variety of features, like code completion, error highlighting, and built-in terminals, which increase productivity and facilitate efficient development.

Preferred Design Tools:

Figma and Adobe XD are the selected design tools. These platforms allow for intuitive UI/UX design creation, collaboration, and testing. With these tools, we can create and iterate upon our application's design in a user-centric manner.

Preferred Quality Assurance Tools:

JUnit and Selenium are chosen for quality assurance. While JUnit provides a robust framework for unit testing our application logic, Selenium provides the capabilities for comprehensive automated UI testing. Both tools will help ensure that the application maintains high quality and stability throughout development and deployment.

Preferred Configuration Management Tools:

Git and Ansible have been selected for configuration management. Git will handle version control, allowing multiple team members to work simultaneously on the codebase with minimal conflicts. Ansible will help automate configuration management tasks, thereby ensuring consistency across development, testing, and production environments.

Preferred Analysis Tools:

Tableau, RapidMiner, and Google Analytics have been selected as the primary analysis tools. These tools will help to analyze user behavior, system performance, and business metrics effectively, guiding future decision-making and iterative improvements.

Preferred Documentation Tools:

Microsoft Word and Confluence are selected for documentation purposes. Word will be used for creating and managing standalone documents, while Confluence will provide a collaborative platform for maintaining project wikis, meeting notes, and team communications. Markdown will also be used for keeping relevant documentation directly alongside the codebase.

Preferred Maintenance Tools:

For maintenance and project management, Jira and GitHub have been selected. Jira will help to manage tasks, bugs, and sprints within the Agile Scrum framework, while GitHub will provide an overview of the codebase and track changes over time.

Preferred Development Technique:

Agile Scrum has been selected as the development technique. This methodology, focusing on iterative progress, collaboration, and customer feedback, is a good fit for the project. It allows for flexibility in development and ensures that the final product closely aligns with user expectations.

The selection of these tools and methodologies will help to ensure that the BAUHINIA Inventory Control Application is developed in an efficient, collaborative, and quality-focused manner, aligning with the user needs and business requirements.

3.3.7 Development of the Business Application

The development of the BAUHINIA Inventory Control Application will follow a three-phase approach: Pre-Development, Development, and Post-Development. Each stage is

crucial to the successful delivery of the application, ensuring the product aligns with user needs and business requirements. Let's delve into what each stage entails.

1) Pre-Development: Planning and Design

This phase lays the groundwork for the entire project. It begins with the identification of user needs and business requirements. The information gathered here will form the basis for the application's functionalities and features.

- Requirements Gathering: This involves consultations with various stakeholders, including business leaders, potential users, and technical teams. The goal is to gain a comprehensive understanding of what the application should achieve.
- Analysis: After gathering requirements, the next step is to analyze these needs and transform them into functional and non-functional requirements. Functional requirements define what the system should do, while non-functional requirements describe how the system should work.
- Design: Based on the requirements, we move on to designing the system. This involves creating various diagrams like ERDs, class diagrams, and sequence diagrams to visualize the application's architecture and data flow. At this stage, we also design the UI/UX of the application using tools like Figma and Adobe XD.

2) Development: Coding and Testing

Once the planning and design are in place, the project moves into the development phase. This is where the actual code is written, and the application begins to take shape.

- Coding: The development team uses PyCharm and Visual Studio Code to write the code in Python, leveraging the Django web development framework. The team will follow best coding practices and standards to ensure the code is efficient, clean, and maintainable.
- Testing: As the application is being developed, it is crucial to test it continually. Using JUnit and Selenium, the team will perform unit testing and UI testing to ensure that all the application components are functioning as expected. In addition

to these, integration tests are performed to ensure different parts of the application work together smoothly.

3) Post-Development: Deployment and Maintenance

After development and testing, the application is ready for deployment. However, the project doesn't end there. Post-deployment, the application needs continuous monitoring, maintenance, and upgrades.

- Deployment: Once thoroughly tested, the application is deployed to the production environment. Here, it becomes available for end users. During deployment, we must ensure that the production environment is correctly configured and that the application performs well under real-world conditions.
- Maintenance: Post-deployment, it's essential to provide ongoing maintenance to the application. This involves troubleshooting any issues, optimizing performance, and ensuring the system's security. Jira and GitHub will be key tools in tracking bugs and managing updates.
- Upgrades: As user needs and business requirements evolve, the application needs to adapt. The team will keep implementing new features, enhancing existing ones, and fixing any bugs that crop up, in line with Agile Scrum's iterative approach.

Through these stages, the BAUHINIA Inventory Control Application will evolve from a concept into a working, useful tool for users. The focus at every stage is to ensure that the final product aligns with user expectations and adds value to the business.

3.4 Support Documentation for the Business Application

The following is a support document intended to guide users through the BAUHINIA Inventory Control Application, including installation and regular use of the application.

3.4.1 User Manual

This section provides an overview of how to install and use the BAUHINIA Inventory Control Application.

Installation Guide:

Installing the BAUHINIA Inventory Control Application involves several key steps:

Step 1: System Requirements

Before installing the BAUHINIA Inventory Control Application, ensure that your system meets the following requirements:

- Operating System: Windows 10 or later, MacOS 10.15 or later
- Web Browser: Google Chrome, Firefox, Safari, or Microsoft Edge latest versions
- Network: Internet connection

Step 2: Download the Application

The application will be provided as a downloadable package. Use the link provided to you by the IT department or your system administrator to download the package.

Step 3: Install the Application

Once downloaded, locate the installation file in your Downloads folder and double click to run it. Follow the instructions presented in the installation wizard. This will

include accepting the terms and conditions, selecting the installation location, and clicking 'Install' to begin the installation process.



Step 4: Launch the Application

After the installation is complete, you can launch the application by finding it in your system's program list or on your desktop, if you chose to create a shortcut.

How-to Guide:

This section provides a basic guide to use the BAUHINIA Inventory Control Application.

a) How to Log In:

Step 1: Launch the application and click on the 'Login' button.

Step 2: Enter your username and password in the appropriate fields.

Step 3: Click 'Submit' to log in to the application.

b) How to Add Items to Inventory:

Step 1: From the dashboard, navigate to the 'Inventory' section.

Step 2: Click on 'Add New Item' and fill in the necessary details such as item name, description, price, and quantity.

Step 3: Click 'Save' to add the item to your inventory.

c) How to Update Item Details:

Step 1: Navigate to the 'Inventory' section from the dashboard.

Step 2: Find the item you want to update and click 'Edit'.

Step 3: Update the necessary details and click 'Save'.

d) How to Generate Reports:

Step 1: From the dashboard, navigate to the 'Reports' section.

Step 2: Choose the type of report you want to generate (Daily Orders Report, Product Availability Report, or Monthly Income Report).

Step 3: Click 'Generate' to produce the report. The report can be viewed, downloaded, or printed for your convenience.

e) How to Track Orders:

Step 1: Navigate to the 'Orders' section from the dashboard.

Step 2: Enter the order ID in the search field and click 'Search'.

Step 3: The order details will be displayed, showing the status of the order.

Remember, for any issues faced during installation or while using the application, please refer to the troubleshooting guide or contact the IT support department.

3.4.2 Technical Documentation

Technical documentation is a comprehensive descriptive and instructional guide meant to assist developers, system administrators, and other technical personnel in understanding the design, functionality, operation, and maintenance of the BAUHINIA Inventory Control Application. It is a critical resource for ensuring efficient use, troubleshooting, and modification of the system when needed.

Code Documentation

Code documentation serves as an essential roadmap for anyone working with the application's codebase, be it new team members getting acquainted with the project, existing team members referencing past work, or future developers seeking to modify or extend the application. Here's a generalized outline of what our code documentation entails:

a) Source File Headers:

Each source file in the BAUHINIA Inventory Control Application's codebase includes a file header comment at the beginning. This header provides an overview of what the file does, who created it, when it was created, and a list of any significant modifications and their dates.

b) Class and Method Documentation:

Every class and method used in the codebase is accompanied by a comment explaining its purpose. These comments detail the class or method's functionality, its inputs and outputs, any exceptions it may throw, and information about the internal workings of the method or class when the functionality isn't self-evident.

c) Inline Comments:

Throughout the code, inline comments are used to describe important or complex code segments. These comments are particularly valuable for enhancing readability and understanding of the code.

d) Naming Conventions and Code Structure:

Our code documentation also emphasizes the use of clear and descriptive naming conventions for all variables, methods, and classes, further enhancing code comprehension. The structure of the code is modular, with each component responsible for a specific subset of functionality, making the code easier to navigate and understand.

e) Code Example:

Below is a hypothetical snippet from our inventory management code, showcasing our documentation approach:

```
 Fibonacci 2.py × Factorial Func 4.py × Coding.py × main.py × Test.py × Fibonacci 3.py × Fibonacci 4.py × Factorial Func 1
1 """"
2 File: inventory.py
3 Created on: 07/2023
4 Author: BAUHINIA Development Team
5
6 This module handles all inventory-related operations for the BAUHINIA Inventory Control Application.
7 """
8
9
10 class InventoryItem:
```

Remember, the actual code of the BAUHINIA Inventory Control Application is far more complex and extensive. This is merely a small, illustrative example of our approach to code documentation.

It's worth noting that we've opted to use tools like Doxygen or Sphinx to automatically generate comprehensive, easily navigable HTML documentation based on these comments and docstrings. This makes it even easier for developers to understand and work with the BAUHINIA Inventory Control Application's codebase.

Database Schema

The database schema serves as the blueprint of our application's database. It defines how data is organized and how different entities within the system are related. Our technical documentation includes a thorough explanation of the BAUHINIA Inventory Control Application's database schema to help developers, data analysts, and other technical personnel understand the structure of the application's underlying data storage.

a) Table Definitions:

The technical documentation provides details about all tables in our database, describing their purpose and all the fields they contain. For each table, we outline its attributes (columns), the data types of these attributes, and any constraints applied to them.

b) Relationships:

The schema documentation also depicts the relationships between different tables. We specify primary and secondary keys and illustrate how tables are connected via these keys. The relationship type (one-to-one, one-to-many, many-to-many) is also clarified for each link between tables.

c) Indexes:

Our documentation also outlines the indexes used in our database. Indexes improve the speed of data retrieval operations on a database table and are crucial in optimizing database performance. We detail what indexes exist, on what attributes they're set, and why they've been created.

d) Views, Stored Procedures, and Triggers:

For any views, stored procedures, or triggers used in the database, we provide comprehensive details. For each, we explain its purpose, what it does, and how it does it, complete with the actual SQL code.

e) Schema Diagram:

To visually represent the information in our database, we include a database schema diagram. This diagram presents tables as blocks with their attributes and shows the connections between these blocks to indicate relationships.

f) Sample Database Schema:

Below is a simplistic example of what part of our database schema might look like:

Table: InventoryItem

- item_id (Integer, Primary Key)
- name (String, Not Null)
- quantity (Integer, Not Null)
- price (Float, Not Null)

Table: SalesRecord

- transaction_id (Integer, Primary Key)
- item_id (Integer, Foreign Key referencing InventoryItem.item_id)
- quantity_sold (Integer, Not Null)
- date_of_sale (Date, Not Null)

This is an extremely simplified version of the database schema for the BAUHINIA Inventory Control Application, serving to illustrate the sort of information included in our actual, much more comprehensive and complex, database schema documentation.

In conclusion, by documenting our database schema thoroughly, we ensure that any team member can understand the structure and function of the database at a glance, facilitating more efficient work and a smoother overall development process.

3.4.3 Conclusion

Summary of the Developed Business Application

The BAUHINIA Inventory Control Application has been designed and developed to meet the needs of businesses looking for a systematic and efficient way to manage their inventory. The application offers functionalities such as stock management, sales tracking, supplier management, and customer information management.

- **Tools and Techniques:** The selection of tools, techniques, and methodologies was based on a thorough investigation. For development, we utilized Python, Django, and SQLite in combination with PyCharm and Visual Studio Code. Microsoft Visio and Draw.io, as well as Figma and Adobe XD, facilitated our design process. Quality assurance was ensured with JUnit and Selenium. Git and Ansible were our configuration management tools. Analysis was conducted using Tableau, RapidMiner, and Google Analytics, and documentation was maintained with Microsoft Word, Confluence, and Markdown. Jira and GitHub were used for maintenance. The chosen development technique was Agile Scrum, which allowed us to be flexible and responsive to changes.
- **Development Stages:** Our development process was broken down into three main stages. Pre-development involved planning and design, including requirements gathering, user flow mapping, and wireframing. The development stage involved

coding and testing, where we implemented the functionality of the application and ensured it worked as intended. The post-development stage covered deployment and maintenance, ensuring the application worked properly in a real-world setting and that any issues that arose were promptly resolved.

- **Support Documentation:** Support documentation was created for both users and developers. The user manual included an installation guide and how-to guide, while the technical documentation included code documentation and a detailed database schema.

Future Recommendations and Enhancements

While the BAUHINIA Inventory Control Application has been designed to be a comprehensive solution for inventory management, there's always room for improvements and enhancements in future iterations. Here are some future recommendations:

- **Mobile Application:** Given the increasing use of mobile devices, a mobile version of the inventory control application would be highly beneficial for on-the-go management.
- **Integration with Other Systems:** We could further enhance functionality by allowing integration with other systems such as accounting software, e-commerce platforms, and CRM systems.
- **Advanced Analytics:** More advanced analytics and reporting features could be added to provide more detailed insights into inventory performance.
- **AI and Machine Learning:** Incorporating AI and Machine Learning capabilities can allow for predictive inventory management, optimizing stock levels, and reducing carrying costs.
- **User Customization:** Allowing more customization options for users would provide a more personalized and efficient experience.

3.5 Assessment of New Ideas and Possible Improvements

The BAUHINIA Inventory Control Application, like any software solution, is not static but an evolving system that can and should be refined over time. Here are a few suggestions for improvements, along with the reasons why they were or were not initially incorporated into the application:

3.5.1 Mobile Application

Potential Improvement:

The demand for mobile applications is escalating in all sectors, and inventory control is no exception. Users nowadays expect their applications to be available anywhere, anytime, on any device. A mobile version of the BAUHINIA Inventory Control Application would cater to this need for flexibility and immediacy.

The potential improvement of a mobile application for BAUHINIA would come with numerous advantages:

- **Increased Accessibility:** With a mobile app, users can access and manage inventory data at their fingertips. This is particularly beneficial for businesses with multiple warehouse locations or those that need to manage inventory on-the-go.
- **Real-Time Inventory Management:** Mobile applications can facilitate real-time inventory tracking, which is crucial for preventing overstocking or understocking situations. Users can receive instant notifications on their mobile devices about low-stock items, pending orders, or other inventory-related alerts.
- **Improved Efficiency:** Mobile applications can streamline operations by allowing users to scan barcodes or QR codes using their mobile device's camera. This eliminates the need for traditional bulky scanning equipment, making the inventory process more efficient and less error-prone.

- Enhanced User Experience: A well-designed mobile application can provide a better user experience with intuitive design and easy navigation. Features such as search, filters, and push notifications can make it easier for users to find and track the information they need.

Justification for Not Including:

However, despite these potential benefits, the mobile application was not included in the initial development phase of the BAUHINIA Inventory Control Application due to several reasons:

- Resource Allocation: Developing a mobile app requires a considerable investment of resources, including time, money, and human resources. The initial focus was on establishing a solid foundation with a web application. Diversifying efforts towards mobile application development in the initial phase could have led to compromises on the quality or functionality of the web platform.
- Skill Set: Mobile app development requires expertise in specific technologies that may differ from those used in web development. Given the team's expertise and familiarity with the chosen web development tools, it was deemed more efficient to leverage these skills for the initial release.
- Complexity: Mobile applications have to account for a variety of device types, screen sizes, operating systems, and versions. This adds an additional layer of complexity to the development and testing process.
- User Base: The need for a mobile app largely depends on the user base's preferences and how they intend to use the system. As the application was in its initial phase, it was important to first understand user behavior and needs on the web platform before extending to mobile.

3.5.2 Integration with Other Systems

Potential Improvement:

System integration is a powerful strategy that can greatly enhance the overall functionality of the BAUHINIA Inventory Control Application. By enabling the application to communicate and share data with other systems like accounting software, e-commerce platforms, or Customer Relationship Management (CRM) systems, the usability and value proposition of the inventory application can be significantly expanded.

- Comprehensive Business Management: Integration with other systems would transform the BAUHINIA Inventory Control Application into a more comprehensive business management tool. For example, integration with accounting software would help automate the financial aspects of inventory management, such as cost calculations, revenue tracking, and financial reporting.
- Streamlined Operations: Integration with e-commerce platforms would automate the synchronization between online sales and inventory levels, reducing the chances of stock discrepancies and enhancing order fulfillment efficiency.
- Improved Customer Service: Integration with CRM systems would facilitate a better understanding of customer buying behavior and preferences, enabling more effective inventory planning and control.
- Enhanced Decision-Making: Integration with other systems would provide a more holistic view of the business, offering valuable insights that can drive strategic decisions regarding inventory management.

Justification for Not Including:

Despite the potential benefits, integration with other systems was not included in the initial development phase of the BAUHINIA Inventory Control Application for the following reasons:

- Complexity: System integration can introduce a significant amount of complexity into the application. Each external system has its own set of APIs, data formats, and

communication protocols. Developing and maintaining the functionality to handle these differences requires substantial time and resources.

- Security: Integrations with external systems often involve the exchange of sensitive data, which necessitates robust security measures to prevent data breaches. Ensuring data security adds another layer of complexity to the development process and requires a specialized set of skills.
- Focus on Core Functionality: The primary goal of the initial development phase was to establish a solid, reliable, and efficient inventory control system. The team decided to focus their resources on this objective before venturing into the added complexities of system integration.
- User Needs: The need for system integration largely depends on the user base's specific needs and how they intend to use the system. It's crucial to have a clear understanding of user needs and behaviors before embarking on complex integration projects.

3.5.3 Advanced Analytics

Potential Improvement:

The inclusion of advanced analytics and reporting features in the BAUHINIA Inventory Control Application represents a significant opportunity for enhancement. These features can provide users with a more comprehensive understanding of their inventory performance and empower them to make data-driven decisions that optimize their inventory management strategy.

- Predictive Analytics: Incorporating machine learning algorithms for predictive analytics could enable users to forecast future inventory needs based on historical data and trends. This could help avoid stockouts or overstocking, both of which carry significant costs.
- Trend Analysis: Tools for identifying and visualizing trends in inventory data over time can help users understand the dynamics of their inventory and make strategic adjustments. For instance, identifying seasonal patterns can inform purchasing decisions and help optimize storage.
- Advanced Reporting: Customizable reports with different viewing options (e.g., by product, by location, over time) would allow users to view their inventory data from various perspectives and dig deeper into the details.
- Performance Metrics: Implementing advanced KPIs like turnover rates, carrying costs, order lead time, and service level can provide valuable insights into the effectiveness and efficiency of inventory management practices.

Justification for Not Including:

- Despite the value that advanced analytics could bring to the application, there were several reasons why this functionality was not included in the initial development phase:
- Resource Constraints: The development of advanced analytics features requires significant time and resources, not only for the development itself but also for ensuring the accuracy and reliability of the analytical models and reports. Given the

constraints of the project, the team decided to focus on the core functionality of the application first.

- User Familiarity: Advanced analytics might introduce a layer of complexity that could be overwhelming for some users, especially those who are not familiar with data analysis. It was important to launch an application that is easy to understand and use for all types of users.
- Evolving Needs: The team decided to start with basic analytics, gather feedback from users, and then gradually add more complex features based on their needs and suggestions. This approach reduces the risk of developing features that users don't find useful.
- Data Availability: Advanced analytics requires a significant amount of data to function effectively. As the application is newly launched, such a volume of data may not be available initially.

3.5.4 AI and Machine Learning

Potential Improvement:

The integration of AI and Machine Learning into the BAUHINIA Inventory Control Application presents a promising frontier for enhancing the application's functionality and value to users. With these advanced technologies, the application could unlock new possibilities for predictive inventory management, thereby significantly optimizing stock levels, reducing carrying costs, and increasing operational efficiency.

- Predictive Inventory Management: Machine Learning algorithms could be trained to analyze historical inventory data, identify patterns and trends, and then make accurate predictions about future inventory needs. This could ensure optimal stock levels at all times, reducing the chances of stockouts or overstocking.
- Demand Forecasting: AI can play a crucial role in demand forecasting by analyzing not just historical sales data but also other external factors like market trends, seasonal variations, and economic indicators. This can lead to more accurate demand predictions and therefore more efficient inventory management.
- Automated Reordering: With AI and Machine Learning, the application could automate the reordering process. When stock levels reach a certain threshold, the system could automatically place orders based on predicted future demand, supplier lead times, and other factors, ensuring continuity of supply.
- Anomaly Detection: AI algorithms can also be used to detect anomalies or irregularities in the inventory data, which could indicate issues like theft, data entry errors, or supplier discrepancies. Early detection of these issues could save the business significant costs.

Justification for Not Including:

Despite the potential benefits of AI and Machine Learning, there were several key reasons why these technologies were not incorporated in the initial version of the BAUHINIA Inventory Control Application:

- Resource Intensive: Developing and implementing AI and Machine Learning capabilities is a complex and resource-intensive task. It requires significant time, specialized expertise, and financial investment, which may not have been readily available during the initial development stage.
- Data Requirements: AI and Machine Learning models require a large volume of high-quality data for training and validation in order to make accurate predictions. As a new application, there may not have been sufficient data available initially to effectively implement these technologies.
- Complexity: Introducing AI and Machine Learning can add a significant level of complexity to the system, both in terms of development and user interaction. For users not familiar with these technologies, this could potentially lead to confusion or misuse.
- Incremental Development: The initial focus was to build a solid foundation with the core features and functionalities. Once this has been achieved and the system is stable, more advanced features like AI and Machine Learning can be considered in subsequent development phases.

3.5.5 User Customization:

Potential Improvement:

Customization of the BAUHINIA Inventory Control Application could potentially be a significant enhancement. It could facilitate a more personalized and efficient user experience, which in turn could drive user engagement and satisfaction. Here are some key aspects where customization could enhance the overall experience:

- **User Interface (UI) Customization:** Allowing users to personalize their UI could help streamline workflows. Users could adjust the layout, choose their preferred themes, toggle between different views, or even select which information should be prominently displayed. This level of customization could make the application more intuitive and easier to navigate, which could boost productivity.
- **Role-Based Access and Customization:** Users with different roles may have different needs. For example, a warehouse manager may want to see different information or have different functionalities compared to a sales representative. By allowing role-based access and customization, each user could tailor the application to their specific needs.
- **Notifications and Alerts:** Giving users the ability to customize their notifications and alerts could ensure that they only receive information that is relevant to them. For instance, they could set up custom alerts for low stock levels, delivery updates, or order deadlines.
- **Reports and Analytics:** Customization in the generation and presentation of reports could allow users to focus on the data that is most important to them. They could define which metrics to track, how often to generate reports, and even how the data is visualized.

Justification for Not Including:

Despite the potential benefits of user customization, there were several compelling reasons for not including it in the initial version of the BAUHINIA Inventory Control Application:

- Development Complexity: Providing a high degree of customization significantly increases the complexity of the development process. Each customizable feature or option needs to be carefully designed and tested to ensure it works correctly in all possible configurations. This could increase the development time, costs, and potential for bugs.
- User Experience (UX) Clarity: While customization can enhance the user experience, it can also complicate it. An overabundance of options can overwhelm users, especially if they're not tech-savvy. For the initial version, it was decided to deliver a clear and straightforward UX, focusing on simplicity and ease of use.
- Stability: In the early stages of development, maintaining system stability is critical. By keeping the system simpler with fewer customizable elements, it's easier to ensure that the system functions correctly and reliably.
- Iterative Approach: The decision was made to use an iterative development approach, focusing first on delivering a solid and stable application with the core functionality. User customization is a feature that can be added in later stages, based on user feedback and needs.

Activity 4

4.1 Critical Review Introduction

4.1.1 A. Brief Overview of the Inventory Control Application

The BAUHINIA Inventory Control Application is a robust, web-based system developed to address the challenges of inventory management for businesses, particularly those within the retail sector. It provides an effective solution for businesses seeking to enhance their inventory control and supply chain efficiency.

The system was conceived and developed to offer functionalities such as tracking inventory levels, orders, sales, and deliveries. It's designed to eliminate manual tracking methods, providing a single, centralized location for all inventory-related data. This significantly reduces the risk of errors associated with manual data handling and ensures that all inventory information is readily accessible and up-to-date.

The application employs a user-friendly interface that supports easy navigation and quick access to features. Built on the Django framework using Python, the system leverages the power of these technologies to offer a robust and scalable solution. Furthermore, the application uses PostgreSQL as its database system to manage and store inventory data.

To support the development process, several tools were employed including PyCharm and Visual Studio Code for coding, Git and Ansible for configuration management, and Jira and GitHub for project management and maintenance. The design process involved the use of Figma, Adobe XD, Microsoft Visio, and Draw.io for creating user interfaces and designing system architecture. Quality assurance was upheld through JUnit and Selenium.

Overall, the BAUHINIA Inventory Control Application is a comprehensive solution aimed at simplifying and optimizing the process of inventory management for businesses. It represents a significant step towards digital transformation in the realm of inventory control and supply chain management.

4.1.2 B. Summary of Problem Definition Statement and Initial Requirements

The Inventory Control Application was conceived and designed to address significant challenges faced by BAUHINIA, a brick-and-mortar retail business looking to expand its presence to the digital space. BAUHINIA had traditionally relied on social media platforms for its online sales and kept physical records of inventory, sales, and customer data, leading to several limitations that hindered its growth and competitiveness in the modern, increasingly digital marketplace.

1) Problem Definition Statement

The primary issue BAUHINIA faced was the need for a comprehensive transformation from their manual, social media-based business operations to a more automated, streamlined, and efficient online platform. This move was crucial to resolving the identified limitations, including restricted customer accessibility, inefficiencies in inventory management due to manual recording, limited reach beyond the local customer base, and the difficulty in leveraging customer data for business insights due to a lack of digitized records.

2) Initial Requirements

To address these challenges and transform BAUHINIA's business operations, specific initial requirements were outlined for the proposed Inventory Control Application.

- Customer Interface: A crucial aspect of the proposed application was a user-friendly customer interface. This interface needed to facilitate smooth customer interactions with the application, such as account creation, login, product browsing, access to detailed product information, and the ability to add items to a shopping cart. It was also essential to incorporate a streamlined checkout process and provide customers with the ability to track their orders.

- Operational Side: On the operational side, the application was required to allow BAUHINIA's staff to manage inventory effectively, generate insightful reports, and have a comprehensive view of business operations through an interactive dashboard.
- Existing System Analysis: The team carried out an in-depth analysis of the existing system at BAUHINIA. This investigation allowed them to identify specific pain points and limitations within the current structure, such as restricted customer accessibility, inefficient inventory management, limited customer reach, and difficulties with leveraging customer data.
- Proposed System Analysis: In response to the existing system's identified limitations, a detailed proposal for a new system was put forward. This proposed system was a comprehensive e-commerce solution designed to streamline BAUHINIA's operations, expand its customer reach, enhance the shopping experience, and harness data for strategic business decision-making.
- User Requirements: User requirements were carefully considered and defined based on the roles and needs of the individual users. These users included BAUHINIA's customers, Inventory Handling Clerks, Production Managers, and the Chief Accountant. Each user's requirements centered on their specific interactions with the application and their role in BAUHINIA's operations.
- System Requirements: System requirements were clearly outlined to guide the application's development. This included both functional and non-functional requirements to ensure the software system could fulfill its intended functions, deliver a seamless user experience, and perform optimally under various conditions.

The initial requirements for the Inventory Control Application were meticulously defined to provide a clear blueprint for the development process. They served as a roadmap for the project, guiding the design, development, and testing of the application. This careful planning was designed to ensure the final application would meet the needs of all users, facilitate smooth and efficient business operations, and adhere to industry standards and

best practices, ultimately enabling BAUHINIA to compete more effectively in the digital marketplace.

4.2 Critical Review of the Design Stage

The design stage of the BAUHINIA Inventory Control Application project involved using various design tools, programming tools, web development tools, configuration management tools, quality assurance tools, documentation tools, design tools, maintenance tools, analysis tools, process modelling tools, prototyping tools, and change control tools. The project was carried out using the Agile Scrum development technique.

4.2.1 A. Recap of the Design Process - Analysis of the Design Tools Used

i. Figma and Adobe XD

As part of the design process, Figma and Adobe XD were chosen as the primary design tools for creating the User Interface (UI) and User Experience (UX) of the BAUHINIA Inventory Control Application. The decision to utilize these tools was guided by their comprehensive suite of features, which made them apt for the design stage. Both applications allowed for real-time collaboration and rapid prototyping, key features which enabled smooth communication and iteration between team members.

Figma and Adobe XD provided the flexibility to design high-fidelity interactive prototypes, which were crucial in obtaining stakeholder feedback and performing user testing. Furthermore, their shared design components and style guides allowed for consistent and scalable designs, which were critical for the application's brand identity. However, despite these strengths, they presented a learning curve that needed to be overcome. Their wide array of features required substantial time for the team to familiarize themselves with, slowing initial progress.

ii. Microsoft Visio and Draw.io

Microsoft Visio and Draw.io were selected for creating diagrams that guided the design and development process. They were specifically used for creating flowcharts, process

diagrams, and network diagrams, which helped visualize the system's architecture and business processes.

Microsoft Visio, with its robust set of features and seamless integration with other Microsoft applications, allowed the team to create complex diagrams with relative ease. Draw.io, on the other hand, offered a simpler, more intuitive interface that was ideal for quick brainstorming sessions and collaborative work.

However, as with Figma and Adobe XD, Visio and Draw.io had their own set of challenges. While Microsoft Visio offered a host of advanced features, it often proved to be overcomplicated for simpler diagrams. Draw.io, despite its simplicity, lacked some advanced features that Visio offered. Balancing the use of these two tools according to the complexity of the task was a learning experience for the team.

In conclusion, while the design tools chosen had their unique strengths, they also presented challenges that required adjustment and learning on the part of the team. Future projects might benefit from a more in-depth preliminary study and training on chosen design tools to expedite the design process.

4.2.2 A. Recap of the Design Process - Effectiveness of the Design created by Designing Tools in Solving the Problem Statement

i. Role of Figma and Adobe XD in Addressing the Problem Statement

The problem statement for the BAUHINIA Inventory Control Application project emphasized the need for a streamlined and efficient system to manage inventory in a user-friendly manner. Figma and Adobe XD, as the primary design tools, were pivotal in creating a solution that addressed this need.

Figma and Adobe XD were used to design an interactive, intuitive, and aesthetically pleasing user interface. This design aimed to simplify complex inventory management tasks, making the system accessible to users with varying levels of technical proficiency. High-fidelity prototypes designed in Figma and Adobe XD allowed for rigorous user testing, enabling us to identify potential issues and rectify them early in the development cycle.

The design process involved consistent user feedback, which played a crucial role in making iterative improvements to the design, thereby enhancing its usability. By enabling real-time collaboration, Figma and Adobe XD empowered the team to work cohesively, ensuring that the design was continually refined to effectively address the problem statement.

ii. Role of Microsoft Visio and Draw.io in Addressing the Problem Statement

Microsoft Visio and Draw.io played an equally important role in the process, helping the team visualize complex system structures and workflows related to inventory management. Using Visio and Draw.io, the team designed diagrams that represented the system architecture, data flow, and business processes.

These diagrams offered a bird's-eye view of the system's functioning, making it easier to identify potential bottlenecks and areas for optimization. By using these tools to plan and design the system's architecture, the team was able to ensure that the final design was robust, scalable, and capable of handling the demands of complex inventory management.

Moreover, these tools facilitated clear and efficient communication between team members and stakeholders. The visual representations created using Visio and Draw.io effectively communicated the design intentions and functionality of the system, which helped in aligning everyone's understanding of the project.

In conclusion, the designs created using Figma, Adobe XD, Microsoft Visio, and Draw.io successfully addressed the problem statement. They helped in crafting a user-centric, efficient, and scalable inventory management system that meets the demands of the modern digital era. The challenge lay in continuously refining these designs, learning from user feedback, and adapting to the evolving project requirements.

4.2.3 A. Recap of the Design Process - Suitability of the Design created by Designing Tools in Meeting Initial Requirements

i. Design Suitability Evaluated through Figma and Adobe XD

The initial requirements for the BAUHINIA Inventory Control Application included creating an easy-to-use, responsive, and visually engaging user interface (UI). The primary design tools chosen for this task were Figma and Adobe XD.

These tools were instrumental in building an intuitive and accessible interface. They allowed for rapid prototyping and design, which meant that UI ideas could be visualized, tested, and iterated on quickly and efficiently. The advanced features offered by Figma and Adobe XD, such as interactive components and auto layout, allowed us to design dynamic, adaptive layouts, which catered to various screen sizes and orientations, thereby fulfilling the requirement for a responsive design.

Furthermore, Figma and Adobe XD also provided powerful user testing and feedback features. Prototypes created using these tools were shared with users and stakeholders, allowing us to gather feedback and make informed adjustments to the design. This iterative process ensured that the design was not only user-friendly but also aligned with the expectations and needs of the end-users, thereby meeting the initial requirement for an easy-to-use interface.

ii. Design Suitability Evaluated through Microsoft Visio and Draw.io

Regarding system architecture and process modeling, Microsoft Visio and Draw.io were the primary tools utilized. The initial requirements for the system included robustness, scalability, and efficient handling of inventory management workflows.

Visio and Draw.io enabled the team to visualize the system's architecture, workflow processes, and data flows effectively. This visualization was key in designing an architecture that was robust and scalable, ensuring that the system could handle high volumes of data and traffic, thereby meeting the initial requirement for robustness and scalability.

The business process and data flow diagrams created using these tools allowed us to study and streamline the inventory management workflows. We could identify potential bottlenecks and inefficiencies in the workflows and address them during the design phase itself, ensuring that the final system was capable of efficiently managing inventory, in line with the initial requirements.

In summary, the design tools used – Figma, Adobe XD, Microsoft Visio, and Draw.io – effectively helped us meet the initial requirements for the BAUHINIA Inventory Control Application. By allowing for rapid, iterative design, visualization of complex system architectures, and workflows, these tools played a crucial role in creating a design that was both user-friendly and capable of handling complex inventory management tasks efficiently.

4.2.4 B. Evaluation of the Design Decisions - Successes and Strengths in the Design Stage created by Designing Tools

i. Successes and Strengths in the UI/UX Design (Figma and Adobe XD)

One of the major successes achieved in the design stage of the BAUHINIA Inventory Control Application was the development of an intuitive and user-friendly interface designed using Figma and Adobe XD.

These tools offered powerful design and prototyping capabilities, which facilitated the creation of a visually appealing and easy-to-navigate interface. We managed to create an application that seamlessly guided users through various functionalities, reducing cognitive load and ensuring a smooth user experience. The collaborative features offered by these tools also proved to be a strength, as they allowed multiple team members to work on the designs simultaneously, enhancing productivity and fostering a cohesive design approach.

The real-time, interactive prototypes created using Figma and Adobe XD allowed us to effectively test the design with end-users and stakeholders. This user-centric design approach led to significant enhancements in the interface's usability, contributing to its success.

ii. Successes and Strengths in the System Architecture and Process Modeling Design (Microsoft Visio and Draw.io)

Another significant success was the design of a robust and scalable system architecture using Microsoft Visio and Draw.io. By leveraging these tools' diagramming capabilities, we could visualize the system's various components and their interrelationships, leading to a solid foundation for the development stage.

The flexibility offered by these tools, coupled with their extensive libraries of shapes and templates, greatly streamlined the process of diagramming the system's architecture and data flows. This efficient process modeling allowed us to anticipate potential bottlenecks and address them proactively, contributing to the system's robustness.

Visio and Draw.io's collaboration features enabled the team to collectively contribute to and review the architectural designs and process flows. This collaboration ensured that the design was consistent and aligned with the team's collective vision and understanding, which was crucial to the success of the design stage.

In conclusion, the design tools Figma, Adobe XD, Microsoft Visio, and Draw.io contributed significantly to the successes and strengths in the design stage. Their powerful design and diagramming capabilities, coupled with their collaboration features, ensured the creation of a user-friendly interface and a robust, scalable system architecture, setting a strong foundation for the development of the BAUHINIA Inventory Control Application.

4.2.5 B. Evaluation of the Design Decisions - Challenges and Areas for Improvement in the Design Stage created by Designing Tools

i. Challenges and Areas for Improvement in the UI/UX Design (Figma and Adobe XD)

Despite the successes achieved in the UI/UX design using Figma and Adobe XD, there were also challenges and areas for improvement. A significant issue faced was the steep learning curve associated with these tools, especially for team members unfamiliar with them. Although the tools are quite user-friendly, mastering the extensive functionalities they offer required considerable time and effort, which somewhat slowed the design process initially.

Additionally, while these tools facilitated the creation of dynamic, interactive prototypes, they do not inherently support user testing. We had to rely on third-party tools to collect feedback and analyze user interactions. The lack of integrated user testing capabilities was a challenge and an area where these tools could improve.

Moreover, there were occasional issues with cross-platform compatibility, especially when working on different operating systems. Some features were not available on certain platforms, causing minor inconsistencies and slowing down the design process.

ii. Challenges and Areas for Improvement in the System Architecture and Process Modeling Design (Microsoft Visio and Draw.io)

The primary challenge in using Microsoft Visio and Draw.io for designing the system architecture and process modeling was their limited support for collaborative work. Although these tools provide basic collaboration features, their real-time co-editing capabilities are not as advanced as some other diagramming tools on the market.

In terms of process modeling, these tools provide a vast library of shapes and templates, but they lack some industry-specific notations and symbols, which sometimes caused difficulties in accurately representing certain aspects of the system architecture or process flow.

Additionally, like Figma and Adobe XD, these tools also presented cross-platform compatibility issues. For example, Microsoft Visio is not natively supported on non-Windows platforms, which presented challenges for team members using different operating systems.

To conclude, while Figma, Adobe XD, Microsoft Visio, and Draw.io brought significant strengths to the design stage, there were challenges that we had to overcome. The steep learning curve, the lack of integrated user testing capabilities, cross-platform compatibility issues, and limited collaborative features are areas where these tools could improve. Nonetheless, the team managed to navigate these challenges and successfully design the BAUHINIA Inventory Control Application. Future design processes could potentially benefit from a more thorough assessment of the chosen tools' limitations, the provision of comprehensive training to the team, and the selection of tools with better collaborative and cross-platform support.

4.3 Critical Review of the Development Stage

4.3.1 A. Recap of the Development Process - Analysis of the Development Tools and Techniques Used

i. Python and Django

Python, a high-level general-purpose programming language, was the primary programming language for the BAUHINIA Inventory Control Application. Known for its simplicity and readability, Python allowed the team to write logic quickly and efficiently. Its extensive standard library and numerous third-party modules made it possible to implement complex functionalities with less code.

Built on Python, Django was used as the web framework for developing the application. Django offers a high-level, MVT (Model-View-Template) architectural pattern, promoting the reusability of components and less code, more flexibility. Its built-in features for handling common web development tasks, like user authentication, form handling, and admin interface, enabled faster and more efficient development. Additionally, Django's ORM (Object-Relational Mapping) system simplified database operations, allowing developers to interact with the database using Python code, instead of writing SQL queries.

ii. PyCharm and Visual Studio Code

PyCharm, a dedicated Python IDE, and Visual Studio Code, a versatile code editor, were chosen as the primary development environments. Both provided a rich set of features including syntax highlighting, code navigation, smart code completion, and debugging tools, which enhanced developer productivity. PyCharm, with its built-in Django support, offered a powerful and intuitive platform for Django development. On the other hand, Visual Studio Code, with its extensive selection of plugins, offered the flexibility to customize the coding environment as per the project needs.

iii. Agile Scrum

The Agile Scrum methodology was selected as the development technique for this project. Agile principles focus on delivering incremental and iterative work packages, which allowed the team to accommodate changes in requirements swiftly. The Scrum approach, characterized by sprints (timeboxed periods for developing a shippable product increment), daily stand-ups, sprint planning, and review meetings, enabled efficient management of the development process. The use of user stories for requirement specification facilitated constant communication and feedback between the team and stakeholders, ensuring that the product evolved to meet the users' needs.

The Scrum approach also promoted a collaborative work environment, where all team members, including developers, designers, and stakeholders, had an active role in the project. The clear definition of roles (Product Owner, Scrum Master, Development Team) and ceremonies (Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective) ensured transparency, regular communication, and continuous improvement.

In summary, Python, Django, PyCharm, Visual Studio Code, and Agile Scrum contributed significantly to the efficient and successful development of the BAUHINIA Inventory Control Application. The chosen tools and techniques offered a blend of simplicity, flexibility, efficiency, and collaboration, enabling the team to deliver a high-quality product in line with user needs.

4.3.2 A. Recap of the Development Process - Effectiveness of the Development Process in Implementing the Design

The effectiveness of the development process is determined by various factors, including the ability to successfully translate the application's design into a functional system, as well as the ability to address the identified problem statement. This involves the interplay of the tools and techniques used, which in the case of the BAUHINIA Inventory Control Application, include Python, Django, PyCharm, Visual Studio Code, and the Agile Scrum development methodology.

i. Python and Django's Effectiveness in Implementing the Design

Python's simplicity and versatility have long been its selling points, and they were particularly effective in this scenario. Its simplicity meant that it was easier to accurately implement the intended design, thereby reducing the chance of errors during coding. Moreover, Python's readability ensured that the codebase remained accessible and comprehensible, promoting better maintenance and future updates.

Django, on the other hand, provided an effective and reliable framework that facilitated the translation of the design into a functional system. Django's architectural pattern, known as the Model-View-Template (MVT) pattern, ensured a clear separation of concerns, which is crucial in maintaining code clarity and organization. This separation made the codebase easier to manage and ensured that changes in one component (e.g., database structure) had minimal impact on the others (e.g., front-end), making the implementation process more efficient and error-free.

ii. PyCharm and Visual Studio Code's Effectiveness in Implementing the Design

The use of advanced Integrated Development Environments (IDEs) such as PyCharm and Visual Studio Code further boosted the effectiveness of the development process. These IDEs come equipped with several features that streamline the coding process and improve

productivity. Features such as intelligent code completion and real-time error detection were instrumental in implementing the design more accurately and quickly, reducing the chances of introducing bugs into the system.

Moreover, their integration with version control systems was particularly beneficial for team collaboration. The ability to track changes and handle merging conflicts efficiently ensured that all team members were synchronized in their efforts, thereby making the development process more cohesive and effective.

iii. Agile Scrum's Effectiveness in Implementing the Design

Finally, the Agile Scrum methodology was another crucial element in ensuring the effectiveness of the development process. Its iterative and incremental approach was instrumental in gradually realizing the design in manageable segments, or sprints. This way, the development team had regular intervals to review and adjust the design implementation, ensuring that the development process was always in line with the design goals.

Agile Scrum also emphasizes constant communication and collaboration, which ensured that discrepancies between the design and implementation were promptly identified and rectified. This, in turn, helped to keep the project on track and ensure the final product met the users' and stakeholders' expectations.

In conclusion, the combination of Python, Django, PyCharm, Visual Studio Code, and Agile Scrum proved highly effective in the development of the BAUHINIA Inventory Control Application. These tools and techniques worked in harmony to facilitate a seamless transition from design to a functional system, thereby successfully addressing the identified problem statement.

4.3.3 A. Recap of the Development Process - Suitability of the Developed Application in Meeting Initial Requirements

The evaluation of the suitability of the developed application against the initial requirements is an integral part of the project review. This process allows us to assess how effectively the chosen development tools and techniques - Python, Django, PyCharm, Visual Studio Code, and the Agile Scrum methodology - were utilized to create a solution that meets the pre-defined requirements.

i. Python and Django: Meeting the Initial Requirements

Python and Django contributed significantly to fulfilling the initial requirements. Python's readability, expressiveness, and extensive library support facilitated efficient and effective coding, helping us realize the functional requirements. Django, being a high-level Python web framework, encouraged rapid development and clean, pragmatic design. Its in-built functionalities, like a templating engine, ORM, and an automatic admin interface, helped to quickly implement required features.

For instance, the Django ORM facilitated interactions with the PostgreSQL database, making it easier to implement inventory data management features in line with the initial requirements. Its security features, such as protection against Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF), ensured that the application met essential security requirements.

ii. PyCharm and Visual Studio Code: Meeting the Initial Requirements

IDEs PyCharm and Visual Studio Code played a crucial role in meeting the project's initial requirements. Both provided an intuitive coding environment with smart code navigation, intelligent code completion, and on-the-fly error checking, which boosted the developers' productivity. These efficiencies directly contributed to meeting project deadlines and fulfilling project requirements.

PyCharm's robust refactoring functionalities were helpful in maintaining a clean and efficient codebase, while Visual Studio Code's extensive extension marketplace allowed developers to add specific features that were useful in achieving certain requirements, such as integration with other tools and technologies used in the project.

iii. Agile Scrum: Meeting the Initial Requirements

The Agile Scrum methodology contributed significantly to the application's suitability. With its iterative nature and short development cycles (sprints), Agile Scrum allowed for regular feedback and adjustment to ensure alignment with the initial requirements.

Each sprint resulted in a potentially shippable product increment, allowing stakeholders to see a working version of the application early and throughout the project. Regular scrum meetings (daily standups, sprint reviews, sprint retrospectives, and sprint planning) enabled constant communication within the team and with the stakeholders, facilitating requirement changes and adjustments to be incorporated swiftly and efficiently.

Furthermore, the use of the Scrum artifact - product backlog, helped in maintaining a clear and prioritized list of requirements. This ensured that the most critical requirements were developed first, enhancing the application's overall value early in the development stage.

In conclusion, the tools and techniques used - Python, Django, PyCharm, Visual Studio Code, and Agile Scrum - significantly contributed to the application's suitability in meeting the initial requirements. Their combined benefits ensured an effective development process, resulting in a functional application that addressed the defined problem statement and fulfilled the project's initial objectives.

4.3.4 B. Evaluation of the Development Decisions - Successes and Strengths in the Development Stage

In the development stage of the BAUHINIA Inventory Control Application, a number of notable successes and strengths emerged. These were largely attributed to the use of Python, Django, PyCharm, Visual Studio Code, and Agile Scrum. Here's a detailed look at each:

i. Successes and Strengths of Using Python and Django

Python's simple syntax, strong integration, and testing capabilities contributed to the smooth development of the application. The project benefited from Python's easy-to-understand and write syntax which enabled rapid coding and quick debugging, speeding up the development process.

Moreover, Django, a Python-based web framework, played a pivotal role in this stage. The framework's 'batteries-included' philosophy meant that it came with various components necessary for web development, which reduced the time spent on boilerplate code. Its Object-Relational Mapper (ORM) simplified database access, while its in-built admin interface significantly eased the process of creating interfaces for admin users.

The use of Django also allowed the team to stick to the Don't Repeat Yourself (DRY) principle, which is one of the core tenets of the framework. This resulted in a more maintainable and less buggy codebase, a significant success for the project.

ii. Successes and Strengths of Using PyCharm and Visual Studio Code

The use of PyCharm and Visual Studio Code as integrated development environments (IDEs) provided a range of useful features that strengthened the development process. The auto-complete and code navigation capabilities of these tools improved the developers' productivity by reducing the time spent on manual code writing and navigation.

Additionally, these IDEs came with built-in debugging tools, which allowed developers to quickly identify and fix errors in the codebase, resulting in a cleaner, more reliable

application. They also offered built-in terminal and command-line interfaces, further accelerating the development process.

iii. Successes and Strengths of Using Agile Scrum

Adopting Agile Scrum methodology was a key strength in the project development stage. This methodology's iterative nature enabled the team to regularly assess the work and adjust as necessary, reducing the risk of project failure. Regular feedback loops ensured that the project remained aligned with the client's expectations and the initial requirements.

Agile Scrum also enhanced team collaboration and communication, contributing to a positive work culture that boosted productivity and motivation. The visibility and transparency that this methodology provided facilitated stakeholder engagement and improved decision-making processes.

In conclusion, the strengths and successes in the development stage can be attributed to the selection of Python and Django as the programming language and framework, the use of PyCharm and Visual Studio Code as IDEs, and the implementation of Agile Scrum as the project management methodology. Together, these tools and techniques created an efficient and effective development process, resulting in a high-quality inventory control application that met the project's requirements.

4.3.5 B. Evaluation of the Development Decisions - Challenges and Areas for Improvement in the Development Stage

In the development stage of the BAUHINIA Inventory Control Application, we encountered several challenges and identified areas for improvement. These revolved around the key tools and techniques used in the project - Python, Django, PyCharm, Visual Studio Code, and Agile Scrum.

i. Python and Django Challenges

Python, recognized for its ease of learning and writing, came with some limitations in terms of execution speed, a significant concern when developing applications requiring fast responses, such as inventory control. Performance issues, notably evident during the testing phase, could cause potential bottlenecks. For future improvements, considering alternatives like PyPy or Cython to boost execution speed or profiling the application for potential performance hotspots could be beneficial.

Django's "batteries-included" approach was indeed handy, providing many prebuilt functions and facilitating the development process. However, it sometimes led to an overhead, especially in smaller applications, and could induce less modular design due to its monolithic nature. One other potential issue lies within Django's ORM (Object-Relational Mapping) that, while making database interactions more Pythonic and easier to handle, may inadvertently create inefficient database queries due to its abstraction of SQL intricacies. To tackle these issues, spending more time on architectural design, ensuring modularity, and studying SQL optimization techniques would be beneficial.

ii. PyCharm and Visual Studio Code Challenges

The integrated development environments (IDEs) - PyCharm and Visual Studio Code - despite their impressive suite of features, sometimes presented performance issues, especially on less powerful hardware due to their heavy resource usage. This could slow down the development process and cause some frustration among the team.

The feature-rich nature of these IDEs also posed a challenge. The sheer number of features, functionalities, and shortcuts required to optimally use these tools had a steep learning curve, potentially slowing down the initial development stages. Ensuring a smooth workflow with various plugins and extensions also required a considerable configuration effort, proving time-consuming.

In response to these challenges, for future projects, a better approach could be to organize dedicated training sessions to ensure the team is well-versed with the IDEs' functionalities and to establish a streamlined setup and configuration process for these development environments.

iii. Agile Scrum Challenges

Agile Scrum, though beneficial in many ways, brought about its own challenges. The methodology requires a high degree of commitment and discipline from the team, something that is not always easy to ensure. It also heavily relies on effective and continuous communication - any breakdown in which could lead to project delays or misaligned deliverables.

Managing the backlog and breaking down larger tasks into manageable sprints sometimes proved challenging, occasionally leading to an uneven distribution of workloads across different sprints. Additionally, the agile nature of the project, while facilitating flexibility, also presented a risk of scope creep due to continuous changes and additions.

To counteract these challenges, implementing more structured planning sessions, introducing stricter controls on backlog changes, and establishing clear communication protocols could be effective strategies for future projects.

Conclusively, even though we faced various challenges during the development stage of the BAUHINIA Inventory Control Application, they were balanced by our successes and learnings. These areas of improvement are not setbacks but valuable lessons that will guide us in refining our development processes for future projects.

4.4 Critical Review of the Testing Stage

4.4.1 A. Recap of the Testing Process - Analysis of the Testing Tools and Techniques Used

The BAUHINIA Inventory Control Application underwent rigorous testing processes to ensure the software system functioned correctly and met all the outlined requirements. Various tools and techniques, including JUnit, Selenium, and Manual Testing, were instrumental in this phase.

i. JUnit

JUnit was the primary tool used for Unit Testing, a critical component of the testing plan. Unit tests were run to evaluate the individual parts of the BAUHINIA system like models, views, and forms, ensuring they performed correctly in isolation. JUnit offered the advantages of readability, simplicity, and swift identification of issues, making it an ideal choice. However, using JUnit mainly involved Java, which could lead to complexities when testing Python-based applications like ours. Thus, we needed to adapt our approach using a similar unit testing framework compatible with Python, such as PyTest or unittest.

ii. Selenium

For Integration Testing and System Testing, Selenium was employed. Selenium facilitated the automated testing of web-based applications, checking the interaction between different parts of the application and the application as a whole. A key strength of Selenium was its capacity to replicate user interactions with the system, including functionalities like sign-up, login, product browsing, and checkout process, among others. However, while Selenium proved beneficial in automating these tests, writing the Selenium test scripts presented a steep learning curve and required substantial time investment. Additionally, it

necessitated a solid understanding of the entire system to ensure all interaction scenarios were adequately covered.

iii. Manual Testing

Manual Testing was employed for Usability Testing, Performance Testing, and Security Testing. Manual testers interacted directly with the software in the same way a user would, checking its user-friendliness, speed and stability under different workloads, and ability to protect against threats. Manual testing provided an intuitive and straightforward way to test the system from the user's perspective, ensuring a smooth and efficient user experience. However, it was also a time-consuming process and prone to human error.

Overall, the choice of tools and techniques used was largely effective, but each came with its own set of challenges and limitations. Future improvements could include leveraging automated testing tools further to reduce manual effort, investing in training for team members on writing test scripts, and incorporating performance and security testing tools for more robust system evaluation.

4.4.2 A. Recap of the Testing Process - Effectiveness of the Testing in Ensuring the Quality of the Application

i. Detecting and Correcting Errors

The testing stage proved effective in detecting and correcting errors in the BAUHINIA system. The testing tools and techniques such as JUnit, Selenium, and Manual Testing enabled the discovery of bugs that could have hindered the functionalities of the software system. Upon identification, these errors were rectified, thereby enhancing the application's overall quality.

ii. Verification and Validation of Functionalities

The testing stage ensured that all functionalities were validated. Functionalities such as sign-up, login, product browsing, and checkout process were subjected to rigorous testing, which allowed the team to confirm that these operations worked as intended. Moreover, this stage confirmed the smooth interaction between different parts of the application, thereby verifying the effective integration of these components.

iii. Usability Testing

The manual testing aspect, specifically usability testing, significantly contributed to ensuring the application's quality. By manually interacting with the system from the user's perspective, the testing team could identify any user-interface issues or bottlenecks that could affect the end-user experience. The feedback obtained in this stage led to significant improvements in the system's user interface and user experience design.

iv. Performance Testing

Performance testing allowed the team to assess the application's stability and speed under varying workloads. This process confirmed that the system could handle the expected user

load and react within acceptable time limits. This validation was crucial in ensuring the application's reliability and responsiveness.

v. Security Testing

With security being a critical aspect of any system, the effectiveness of testing in this area was crucial. Manual security testing was performed to identify any potential threats or vulnerabilities. The application was tested for common web application vulnerabilities, such as SQL injection and Cross-site Scripting (XSS), ensuring that it could withstand potential security attacks.

Despite the successes of the testing stage, some areas could be improved. The manual testing process, although vital, proved time-consuming and required significant human resources. Automated testing tools could be used more extensively in future iterations to improve efficiency. Additionally, while Selenium was effective in integration and system testing, the time and expertise needed to write Selenium test scripts was considerable. Providing more training to team members could mitigate this issue.

Overall, the testing process effectively ensured the quality of the BAUHINIA Inventory Control Application. It verified the proper functioning of all the components, ensured user-friendliness, confirmed system performance under workload, and validated the system's security measures.

4.4.3 A. Recap of the Testing Process - Suitability of the Testing Process in Validating the Initial Requirements

i. Verifying Functional Requirements

The testing process played a crucial role in verifying the initial functional requirements. Each feature of the BAUHINIA system, including functionalities related to Customer and Staff pages like sign-up, login, product browsing, checkout, shopping cart, order tracking, reports, inventory update, and dashboard, were tested against the predefined requirements. This thorough process ensured that all functional requirements were met, enhancing the system's value to end-users.

ii. Confirming Non-Functional Requirements

Non-functional requirements, such as performance, security, and usability, were also thoroughly validated during the testing process. Performance testing confirmed that the BAUHINIA system could handle the expected load and respond quickly to user requests. Security testing ensured that the application was capable of protecting against potential threats. Usability testing was performed to ensure the system was user-friendly and met the requirements for a smooth and intuitive user interface.

iii. Testing for Edge Cases and Exception Handling

The testing process also included testing for edge cases and exception handling, which are often overlooked during the initial requirement gathering process. By identifying how the system responded to unexpected or extreme inputs, the testing team was able to ensure that the application could handle exceptions gracefully and effectively.

iv. Iterative Testing

The iterative nature of the testing process was also beneficial in validating the initial requirements. As testing was carried out in parallel with development, it provided early

feedback on any discrepancies between the application's behavior and the initial requirements. This allowed the development team to quickly correct course and adhere to the initial plan.

v. Alignment with Agile Scrum Methodology

Using Agile Scrum as the project management approach allowed for regular validation of the requirements throughout the sprints. Each sprint ended with a testing phase where the implemented features were tested against the initial requirements, ensuring alignment at every step of the development process.

While the testing process proved effective in validating the initial requirements, there is always room for improvement. More automated testing could be implemented to speed up the process and reduce the risk of human error. Additionally, a more detailed requirements analysis stage could help clarify complex requirements, reducing the possibility of misunderstandings that need to be rectified during testing.

In summary, the testing process was largely successful in validating the initial requirements of the BAUHINIA system. By focusing on verifying both functional and non-functional requirements, handling edge cases, and leveraging the iterative nature of Agile Scrum, the testing team could ensure that the system adhered closely to the initial plan.

4.4.4 B. Evaluation of the Testing Decisions - Successes and Strengths in the Testing Stage

i. Comprehensive Testing Approach

One of the notable strengths in the testing stage was the adoption of a comprehensive testing approach. The BAUHINIA system was subjected to various types of testing, including Unit Testing, Integration Testing, System Testing, Usability Testing, Performance Testing, and Security Testing. This holistic strategy helped uncover a wide array of potential issues and ensured that all aspects of the system were rigorously evaluated, significantly improving the quality of the final product.

ii. Integration of Manual and Automated Testing

The testing stage successfully combined the benefits of both manual and automated testing. Tools like JUnit and Selenium were employed for automating repeatable tests and improving the efficiency of the testing process. Manual testing was also used, particularly for usability testing and complex scenarios that require human judgment. This balanced approach optimized the strengths of each testing method and reduced their inherent weaknesses.

iii. Effective Alignment with Agile Scrum Methodology

The testing decisions were also effectively aligned with Agile Scrum methodology. This allowed the testing team to work in tandem with the development team, enabling early detection and rectification of defects, thereby reducing the overall project risks. This iterative approach to testing also ensured that the system was continuously evaluated against the initial requirements throughout its development.

iv. Thorough Documentation

Another strength of the testing stage was the thorough documentation of the test plan, including well-defined features to be tested, test cases, and the criteria for passing or failing a test. This clarity and transparency in the testing process enhanced the traceability of the testing activities and allowed for effective communication of the testing progress to all stakeholders.

v. Risk and Contingency Planning

Risk and contingency planning were also well-implemented during the testing stage. Potential software or hardware failures, resource availability, and schedule adherence were identified as risks, and appropriate contingency plans were put in place. This proactive approach helped to minimize disruptions during the testing stage and ensured the timely completion of the testing activities.

In conclusion, the testing stage for the BAUHINIA system demonstrated several key strengths and successes. The comprehensive and balanced testing approach, effective alignment with Agile Scrum methodology, thorough documentation, and proactive risk management contributed significantly to the successful delivery of a high-quality system that met the initial requirements.

4.4.5 B. Evaluation of the Testing Decisions - Challenges and Areas for Improvement in the Testing Stage

i. Time Constraints and Resource Limitations

While the testing stage was generally successful, there were challenges related to time constraints and resource limitations. The comprehensive nature of the testing strategy meant that a significant amount of time was needed to execute all planned test cases. In future projects, considering more efficient ways to allocate resources, such as parallel testing or more effective automation strategies, could help mitigate these challenges.

ii. Limitations of Automated Testing

Despite the benefits of automated testing, certain limitations became apparent during this stage. Automated tests, though fast and efficient for certain repetitive tasks, can miss out on issues that manual testing could have caught. A possible area for improvement could be allocating more time for manual exploratory testing, particularly for the user interface and usability aspects that require human judgment.

iii. Coverage of Test Cases

Although a diverse range of test cases was used, ensuring the coverage of all possible scenarios is always a challenge in software testing. While the major functionalities were thoroughly tested, there may have been some edge cases or unusual user behaviors that were not covered by the test cases. For future improvement, investing more time in the creation of test cases and considering techniques like pairwise testing or state transition testing might help to ensure a more comprehensive coverage.

iv. Handling of Test Data

The management of test data, especially when dealing with a large application like the BAUHINIA system, was a challenging task. For future projects, more focus could be given

to test data management strategies and the use of dedicated tools to automate the process of creating, managing, and disposing of test data.

v. Early Integration of Performance and Security Testing

Performance and security testing were conducted, but integrating these tests earlier into the development lifecycle could be beneficial. By doing this, any issues related to performance or security can be detected and addressed sooner, reducing the risk and potential impact on the project.

Overall, while the testing stage demonstrated a number of successes, these challenges and areas for improvement provide valuable lessons for future projects. By addressing these points, the efficiency, effectiveness, and comprehensiveness of the testing process can be further enhanced, contributing to an even higher quality of the final product.

4.4.6 C. Evaluation of the Test Plan and Test Cases - Test Plan Assessment

i. Strengths of the Test Plan

The Test Plan was designed meticulously to cater to all features and functionalities of the BAUHINIA system.

- Comprehensive Coverage: The Test Plan outlined a comprehensive list of features to be tested, covering everything from simple login functionality to more complex tasks like report generation and inventory updates. This ensured a broad scope of testing, increasing the chance of catching and rectifying potential issues before deployment.
- Variety of Testing Techniques: The Test Plan incorporated a variety of testing techniques, including Unit Testing, Integration Testing, System Testing, Usability Testing, Performance Testing, and Security Testing. This variety allowed for the robust evaluation of the application from multiple angles, thus ensuring thorough validation.
- Detailed Test Cases: The Test Plan included a specific set of test cases for every functionality. Each test case was designed to ensure that the application responded correctly to a variety of scenarios, including edge cases.
- Clear Pass/Fail Criteria: The Test Plan outlined clear pass/fail criteria, which helped in maintaining objective and standardized testing. It was thus easy to identify when a test case had failed and needed further investigation.

ii. Challenges of the Test Plan

Despite its strengths, the Test Plan also encountered certain challenges.

- Time Management: Given the comprehensive nature of the Test Plan, executing all test cases within the stipulated timeline proved to be a challenge.
- Resource Allocation: There was a need to balance between manual and automated testing resources. Too much reliance on either could lead to inefficiencies and missed defects.

- Handling Test Data: Creating realistic and effective test data was a challenge, particularly for complex functionalities.

iii. Areas for Improvement

Some areas for improvement in the Test Plan include:

- More Efficient Test Strategies: Implementing more efficient test strategies, such as risk-based or priority-based testing, could help manage time and resources more effectively.
- Better Test Data Management: There could be a better strategy for test data management, perhaps through the use of dedicated tools or services, which can help in creating, managing, and disposing of test data.
- Earlier Integration of Non-functional Testing: Integrating non-functional testing like performance and security testing earlier in the development cycle can help in identifying potential issues sooner.
- Inclusion of Negative Testing: The Test Plan could incorporate more negative test cases to ensure the system behaves correctly even when fed with incorrect, unexpected, or null input.

In conclusion, the Test Plan was comprehensive and effective, but there's always room for improvement. Learning from the challenges encountered, future Test Plans could become more efficient and effective, contributing to the overall quality and reliability of the software.

4.4.7 C. Evaluation of the Test Plan and Test Cases - Test Cases Assessment

i. Strengths of the Test Cases

The Test Cases were a key part of the testing strategy, and their design played a significant role in the successful testing of the BAUHINIA system.

- Comprehensive and Detailed: The Test Cases covered a wide range of functionalities in the system. Each test case was explicitly described, ensuring a thorough understanding of the steps to be performed, the expected result, and the actual outcome. This detailed approach aided in identifying any discrepancies between expected and actual results.
- Inclusion of Real-world Scenarios: The test cases incorporated real-world scenarios, which helped in assessing how well the application performed under conditions similar to those it would face after deployment. This contributed to the validation of the application in a practical context.
- Emphasis on Core Functionalities: The Test Cases paid special attention to the system's core functionalities, like product browsing, order tracking, and inventory updates. This helped ensure these essential features worked as expected.

ii. Challenges of the Test Cases

While the Test Cases were overall successful, some challenges were experienced in the process.

- Coverage of Edge Cases: Covering all possible edge cases proved to be a challenge. While many were considered, it's virtually impossible to predict all potential anomalies that could occur when the application is used in a real-world environment.
- Time and Resource Intensive: The creation, execution, and maintenance of the test cases required a significant investment of time and resources. In some cases, time constraints could lead to rushing through some test cases, potentially missing out on some issues.

iii. Areas for Improvement

There were a few areas where the Test Cases could be improved for future projects.

- Automating Test Cases: For repetitive and data-intensive test cases, automation could save time and resources while increasing accuracy. Tools like Selenium can help automate these tasks and make the process more efficient.
- Prioritizing Test Cases: In order to use resources more effectively, test cases could be prioritized based on the feature's importance, complexity, and risk factor. This would ensure that key functionalities are thoroughly tested, even when time or resources are limited.
- Incorporating More Negative Test Cases: More emphasis could be placed on creating negative test cases to validate how the system handles errors or unexpected inputs. This would help identify any potential issues or weaknesses in error handling and input validation.

In conclusion, while the Test Cases were quite effective in guiding the testing process and identifying issues, there are always areas for improvement. By refining the test case design process and adopting more efficient strategies, the quality of testing can be further enhanced, leading to more reliable software.

4.5 Critical Review of the Business Application

➤ Recap of Problem Definition Statement and Initial Requirements

In the initial stages of the project, the problem definition statement and initial requirements were established to create a clear focus for the project team. The problem statement identified the need for an efficient, effective, and user-friendly e-commerce system called BAUHINIA. This system aimed to cater to both staff and customer needs, covering areas such as product browsing, order placement, inventory management, and report generation.

The initial requirements laid out specific functionalities to solve this problem. These included creating a smooth and easy user interface, enabling secure sign-up and login functionalities, offering comprehensive product browsing and detail views, managing a shopping cart system, enabling secure and efficient checkout processes, tracking orders, creating reports, and managing inventory effectively.

4.5.1 A. Evaluation of the Business Application Performance against the Problem Definition Statement

Analyzing the performance of the BAUHINIA system in relation to the problem definition statement, it is evident that significant strides have been made in addressing the issues outlined. The problem definition statement originally identified the need for an effective, efficient, and user-friendly e-commerce platform catering to both customers and staff. Let's examine how well the BAUHINIA system has been able to address these areas:

- **Effective System:** The BAUHINIA system has been designed and developed to manage the complexities of e-commerce effectively. With distinct modules to handle different aspects such as customer orders, product browsing, inventory management, and reporting, the system is well-equipped to meet the challenges of an e-commerce business. It has successfully incorporated a wide range of functionalities that align well with the typical needs of an online business, thereby effectively addressing the problem definition statement.

- Efficient System: Efficiency was a core aspect of the problem definition, and the BAUHINIA system has managed to deliver on this front as well. The use of Python and Django has facilitated the creation of an application that is not just robust but also highly efficient. Order processing, report generation, and inventory management are handled seamlessly, reducing manual intervention and associated errors. This results in a significant enhancement of the overall business process efficiency, in line with the requirements of the problem definition statement.
- User-Friendly: User-friendliness was a critical requirement, especially considering the two types of users - the customers and the staff. The BAUHINIA system has incorporated user-friendly design principles to deliver a smooth and intuitive user experience. This is apparent in the simple, clean design of the interface, the ease of navigation, and the straightforward process flows for order placement, tracking, and inventory management. Both customer and staff portals are easy to use and intuitive, providing a user-friendly solution that matches the problem definition statement.

In summary, the BAUHINIA system has successfully addressed the key components of the problem definition statement. It offers an effective, efficient, and user-friendly solution for an e-commerce platform that caters well to the needs of both customers and staff. This shows that the development process has been in line with the problem statement, and the resulting application provides a comprehensive solution to the issues identified initially.

4.5.2 A. Evaluation of the Business Application Performance against the Initial Requirements

The initial requirements of the BAUHINIA system revolved around building a comprehensive and intuitive e-commerce platform that would cater to both customers and staff members with distinct sets of functionalities for each. Let's examine how the final application performed against these initial requirements:

a. Customer Features:

- Account Registration & Login: The system effectively allows customers to create an account and log in securely. This feature was implemented successfully and provides an excellent user experience, meeting the initial requirements.
- Browsing Products: Customers can smoothly browse through available products, with each product's essential details readily accessible. This feature has surpassed the initial requirements, providing an engaging and user-friendly browsing experience.
- Shopping Cart: The shopping cart functionality is implemented effectively, allowing customers to add, update, or remove items from their cart before finalizing the purchase.
- Checkout & Payment: The checkout and payment process is seamless, user-friendly, and secure, fulfilling the initial requirements.
- Order Tracking: The tracking system allows customers to check the status of their orders, providing a vital feature that meets the initial requirements.

b. Staff Features:

- Inventory Management: The BAUHINIA system provides staff with the capability to manage inventory effectively, update product information, and keep track of stock levels. This feature has been implemented successfully, addressing the initial requirements.

- Order Processing: The system facilitates efficient order processing, allowing staff to update order statuses and manage dispatch and delivery, aligning with the initial requirements.
- Reports Generation: The staff can generate various reports, providing key insights into sales, customer behavior, and inventory status. This feature is in line with the initial requirements.
- Dashboard: The staff dashboard provides an overview of the ongoing operations, assisting in better decision-making. This requirement was met successfully.

- Customer Requirements: The initial requirements included various functionalities for customers, such as account creation, browsing products, adding items to the cart, placing orders, and tracking their orders. The BAUHINIA system has successfully incorporated these features, providing a seamless shopping experience for the customer. Moreover, the clean, intuitive user interface promotes easy navigation and makes the overall experience hassle-free, thereby satisfying the initial requirements.
- Staff Requirements: For staff members, the initial requirements comprised functionalities such as inventory management, order processing, and report generation. In the BAUHINIA system, staff members have a separate dashboard from where they can manage inventory, process orders, and generate different types of reports. Thus, the system caters to the initial requirements by providing a platform that streamlines staff workflows and enhances productivity.
- Performance Requirements: In terms of performance, the system was required to be robust, secure, and fast, providing a smooth user experience. Through efficient use of Python and Django, along with systematic testing and debugging, the BAUHINIA system offers high performance and security. The system handles multiple simultaneous requests efficiently and maintains quick response times, meeting the initial performance requirements.
- Security Requirements: Security was a crucial aspect of the initial requirements, especially considering the sensitive user information the system handles. The

BAUHINIA system has integrated robust security measures, including secure login mechanisms, encrypted data transmission, and secure handling and storage of user data. This commitment to security matches the initial requirements, ensuring users' trust and confidence in the system.

In summary, the BAUHINIA system appears to meet and, in some areas, exceed the initial requirements set for the project. It provides a comprehensive solution that caters to the diverse needs of an e-commerce platform. However, continuous refinement and feature additions will further enhance the system's overall performance and user experience.

4.5.3 B. Evaluation of the Business Application Strengths and Weaknesses

1. Strengths of the Business Application

- User-Friendly Interface: One of the key strengths of the BAUHINIA system is its user-friendly interface. The design is intuitive, making it easy for both customers and staff to navigate and complete their tasks.
- Robust Functionality: The system is equipped with a wide array of features and functionalities, covering a comprehensive spectrum of operations from customer sign-up to order tracking and inventory management. This provides a single integrated platform for users to manage their interactions with the business.
- Responsive Design: The system's responsive design allows it to be accessed and used seamlessly across various devices and screen sizes. This enhances user experience, especially in today's digital age where people use different devices for accessing online platforms.
- Secure and Reliable: The system has been developed with a focus on security, which is crucial in today's cyber-threat environment. Security testing was carried out to ensure the system is robust against threats.
- Efficient and Scalable: Performance testing has shown that the system can handle a significant workload without any notable decrease in speed or stability. Moreover, the system architecture supports scalability, which means it can accommodate the growth of BAUHINIA's business.

2. Weaknesses of the Business Application

- Limited Automated Testing: Although the system was tested rigorously, the testing was largely manual, which is time-consuming and susceptible to human error. Automated testing could have increased efficiency and accuracy.
- Complexity of Features: While the system is packed with features, the sheer number and complexity of these features might be overwhelming for some users, especially those who are not tech-savvy.
- Dependency on Third-Party Services: There are certain dependencies on third-party services, such as payment gateways, which can introduce external risks and potential points of failure.
- Requirement of Continuous Updates: With the rapid pace of technological advancement, the system needs regular updates to stay relevant and secure. This continuous need for maintenance can be a burden.
- Lack of Real-Time Inventory Update: Currently, inventory updates are carried out manually by the staff, which can lead to delays and inaccuracies. Implementing real-time inventory updates could enhance operational efficiency.

In summary, while the BAUHINIA system has numerous strengths that make it a powerful business application, there are also weaknesses and areas for improvement that need to be addressed to ensure its long-term success and sustainability.

4.6 Recap of Initial Risks Identified

4.6.1 A. Brief Summary of the Initial Risks

At the outset of this project, a variety of potential risks were identified, spanning different areas of the project. These were:

- Technical Risks: These were related to the potential for encountering difficulties with new technologies or tools, data migration from the old system, or issues integrating the new system with existing ones.
- Project Management Risks: These involved possible delays due to planning challenges, miscommunication within the team, or the phenomenon of scope creep, where new features are continuously added to the project, causing it to deviate from the original plan.
- Financial Risks: These revolved around potential financial challenges such as budget overruns due to unforeseen costs, or not achieving the desired return on investment if the system failed to meet its objectives or was not adopted as expected by the users.
- Operational Risks: These included potential challenges that could arise during the system's operational life, such as system downtime, performance issues, or maintenance difficulties.
- Legal and Compliance Risks: These referred to possible legal issues or non-compliance with regulations, like data protection laws.
- Security Risks: These involved potential vulnerabilities and threats that could compromise the system's security, such as data breaches, unauthorized access, or other forms of cyberattacks.
- Usability Risks: These were related to the possibility of the system being difficult to use or not meeting user expectations.
- Vendor and Third-party Risks: These involved potential risks related to dependencies on external parties like software vendors or service providers.

- Organizational Change Risks: These included the potential challenges associated with introducing a new system within the organization, such as resistance from employees or difficulties in training them to use the new system.
- Scalability Risks: These involved potential difficulties related to the system's ability to scale and handle increased user load or data volume.

4.6.2 B. Evaluation of How Risks Were Addressed or Materialized During Development

During the application development process, the different categories of risks were managed, minimized, or materialized in the following ways:

- Technical Risks: The development team encountered a few technical challenges during the implementation of the project, such as the integration of new technologies and data migration. However, these issues were efficiently managed by leveraging the team's technical expertise and utilizing a problem-solving approach that prioritized effective communication and collaboration. The team's adaptability and commitment to continuous learning were key in overcoming these risks.
- Project Management Risks: Effective project management methodologies like Agile Scrum were used, which helped keep the project on track and ensured that potential risks associated with scope creep, miscommunication, and delays were minimized. Regular team meetings and status updates ensured everyone was aligned, and any deviations from the original plan were quickly identified and rectified.
- Financial Risks: The project adhered closely to the pre-defined budget, with any potential overruns identified early through regular financial monitoring. Through judicious resource allocation and continuous financial oversight, the project was completed without major financial issues, mitigating the financial risks.
- Operational Risks: Extensive system testing was done to mitigate operational risks like system downtime or performance issues. The system was evaluated under different conditions to ensure it performed optimally and reliably. Post-deployment,

continuous monitoring has been set up to identify and resolve any operational issues swiftly.

- Legal and Compliance Risks: The development team strictly adhered to all relevant data protection laws and regulations throughout the development process. Regular compliance checks were carried out to ensure that the system met all legal requirements, thus successfully managing legal and compliance risks.
- Security Risks: Security was a priority throughout the development process. The team incorporated secure coding practices and conducted multiple rounds of security testing to identify and resolve potential vulnerabilities. Regular updates and patches are planned to ensure continued security of the application.
- Usability Risks: User testing and feedback were integral parts of the development process. These helped identify potential usability issues early in the development process, and features were iteratively improved for optimal user experience. Feedback was actively sought and incorporated, ensuring that the system met and exceeded user expectations.
- Vendor and Third-party Risks: The team ensured that all third-party service providers and vendors used in the project were reliable and had strong track records. Any issues that arose were resolved quickly in collaboration with the vendors.
- Organizational Change Risks: An effective change management strategy was employed to manage the transition to the new system. This included comprehensive training for employees, clear communication about the changes, and ongoing support to address any issues or concerns.
- Scalability Risks: The system was designed and tested for scalability from the outset. It has been built to handle increased user load and data volume, ensuring that it will continue to perform effectively as BAUHINIA's business grows.

In conclusion, each of the potential risks identified at the beginning was effectively managed and mitigated. The team's proactive approach to risk management contributed significantly to the successful delivery of the application. However, risk management is an ongoing process, and the team will continue to monitor and address any new risks that may arise during the operational phase of the system.

4.6.3 C. Evaluation of Risk Identification Effectiveness

- Effectiveness of Initial Risk Identification: The initial risk identification process was highly effective, as it provided a comprehensive overview of potential challenges that could arise during the development and implementation of the application. This process involved anticipating risks across multiple categories, including technical, project management, financial, operational, legal, security, usability, vendor, organizational change, and scalability risks. By outlining these risks upfront, the team was able to develop strategies to mitigate them effectively.
- Proactiveness and Preparedness: The team's proactive approach to risk identification significantly enhanced its preparedness for dealing with potential challenges. The identified risks served as a roadmap, helping the team anticipate problems and take preventive action. Moreover, a risk mitigation plan was established for each identified risk, which provided clear guidance on how to handle the risks should they materialize.
- Effectiveness in Practice: The risk identification process proved effective in practice as well, as most of the anticipated risks were successfully managed and mitigated. In some cases, the risks did materialize, such as technical and project management risks. However, due to the upfront identification of these risks and subsequent planning, the team was able to address them promptly and minimize their impact on the project.
- Areas for Improvement: While the initial risk identification was largely effective, there is always room for improvement. It would be beneficial for future projects to incorporate a more systematic risk identification process that also considers less obvious or secondary risks. Moreover, involving a wider range of stakeholders in the risk identification process could help bring in diverse perspectives and uncover risks that may not have been considered initially.

In conclusion, the initial risk identification process was highly effective in anticipating potential challenges and preparing the team to address them. This proactive approach played a significant role in the successful development and implementation of the business

application. Future projects could benefit from a more comprehensive and systematic risk identification process, which considers a broader range of risks and involves a wider range of stakeholders.

4.6.4 D. Analysis of Achievements and Strengths in Risk Identification and Mitigation

- Proactive Risk Identification: A significant success in the project was the team's proactive approach to identifying potential risks. The team thoroughly considered various types of risks - technical, project management, financial, operational, legal, security, usability, vendor, organizational change, and scalability - that could impact the project. This proactive risk identification allowed the team to anticipate potential problems and formulate mitigation strategies in advance.
- Comprehensive Risk Mitigation Plans: The development and implementation of comprehensive risk mitigation strategies were notable strengths of the project. For each risk identified, the team developed a clear plan of action to minimize its impact if it were to materialize. These plans were regularly updated and refined as the project progressed, which ensured their effectiveness.
- Effective Communication and Collaboration: The team successfully leveraged effective communication and collaboration to manage and mitigate risks. Information about potential risks and their mitigation plans was shared with all relevant stakeholders, ensuring everyone was aware of the possible challenges and their solutions. This open communication fostered a collaborative environment, enabling the team to work together effectively in addressing risks.
- Successful Risk Management in Practice: The risk identification and mitigation strategies were not just theoretical; they proved successful in practice as well. When some of the identified risks materialized, the team was prepared and acted swiftly to minimize their impact. This demonstrates the practical effectiveness of the risk management approach adopted by the team.
- Lessons Learned and Continuous Improvement: The team took the approach of treating risks and their management as opportunities for learning and improvement. Whenever a risk materialized, the team analyzed the situation, evaluated their

response, and took note of lessons learned. This reflective approach helped the team improve their risk management skills continuously and will be valuable for future projects.

In summary, the proactive and systematic approach to risk identification, the comprehensive risk mitigation plans, the effective communication, the successful management of risks in practice, and the continuous learning and improvement mindset were notable strengths and successes in the risk identification and mitigation process of the project.

4.7 Opportunities for Improvement

4.7.1 A. Identified Areas for Enhancement

1. Enhancing Usability and User Experience:

- Simplified Navigation: The peer review feedback highlighted potential challenges users may face in navigating the application. Thus, efforts can be made to redesign the navigation menu for ease of use, possibly by grouping related functionalities together or by introducing a search functionality for quicker access.
- UI Refinements: A more visually appealing and intuitive user interface could further improve the user experience. This could involve using user-centric design principles, leveraging color psychology, and ensuring consistency across all pages of the application.
- Performance Optimization: Slow loading times can hamper user experience. Thus, optimization techniques, such as code optimization, lazy loading, and caching, can be implemented to speed up the application.

2. Feature Enhancement:

- Advanced Analytics: Currently, the application provides basic inventory management functionalities. However, integrating advanced analytics could enable the generation of valuable insights, such as prediction of inventory requirements, identification of slow-moving items, and trend analysis.
- Comprehensive Reports: While the application allows basic reporting, there's room for introducing more comprehensive and detailed sales and inventory reports, featuring visualizations for easy interpretation of data.

3. Integration Capabilities:

- Interoperability: The application's capability to interact with other systems is critical for smooth business operations. Enhancements could involve developing APIs for

easier data exchange with other platforms or integrating middleware for real-time data syncing.

- Single Sign-On: To reduce the burden of managing multiple credentials and to enhance user experience, a Single Sign-On (SSO) feature could be introduced to allow users to access different systems within the BAUHINIA ecosystem seamlessly.

4. Security Enhancements:

- Data Encryption: To protect sensitive business data, stronger encryption algorithms could be used. The introduction of end-to-end encryption ensures data is unreadable to unauthorized users, thereby providing an extra layer of security.
- Multi-Factor Authentication (MFA): For further strengthening user authentication, MFA could be introduced. MFA adds an extra step to the login process, thereby making it harder for malicious actors to gain access.

5. Scalability:

- Cloud Technologies: To ensure the application can handle a growing user base and data load, cloud-based technologies could be employed. The use of cloud-based solutions could offer flexible storage options, cost efficiencies, and easier scaling.
- Distributed Systems: By leveraging a distributed system, where components are located on networked computers, the application can achieve better scalability, performance, and reliability.

6. Mobile Compatibility:

- Responsive Design: To cater to users accessing the application from various devices, a responsive design can ensure the application adapts to different screen sizes, providing a seamless user experience.

- Native Mobile Applications: Depending on user needs and business goals, native mobile applications for Android and iOS platforms could be developed, providing users with the flexibility to manage inventory on the go.

7. Automated Testing:

- Tool Implementation: The adoption of automated testing tools can increase the speed of testing cycles, enable parallel execution, and eliminate human errors.
- Continuous Integration and Continuous Deployment (CI/CD): Implementing a CI/CD pipeline can help in automating the testing process, ensuring every code change is automatically tested before it is deployed to the production environment. This would enable quicker detection and resolution of bugs, thereby improving the application's quality and reliability.

The feedback and reflections garnered from the development process have not only shed light on the strengths of the BAUHINIA Inventory Control Application but also have identified considerable opportunities for improvement. This iterative approach of development, feedback, and refinement is critical to continually enhance the application and cater to the evolving needs of BAUHINIA and its customers.

4.7.2 B. Justification for Each Suggested Improvement

1. Enhancing Usability and User Experience:

- Simplified Navigation: Effective navigation is the cornerstone of usability. Clear, logical, and intuitive navigation ensures users can quickly find what they need, leading to higher user satisfaction and productivity.
- UI Refinements: The user interface serves as the bridge between users and the application. An attractive, easy-to-use interface promotes user engagement, reduces learning curve, and enhances user satisfaction.
- Performance Optimization: A slow application can lead to user frustration and potential loss of users. By enhancing performance, we ensure that users can perform tasks quickly and efficiently, thereby increasing their productivity.

2. Feature Enhancement:

- Advanced Analytics: With advanced analytics, BAUHINIA can make data-driven decisions. It helps identify trends, uncover insights, and predict future inventory needs, thereby aiding strategic decision-making and reducing the risk of overstocking or understocking.
- Comprehensive Reports: Detailed reporting is crucial for understanding the inventory status, making forecasts, and identifying issues. Visual reports can communicate complex data in an understandable way, aiding faster and more accurate decision-making.

3. Integration Capabilities:

- Interoperability: Integration with other business systems ensures consistent and up-to-date data across platforms, reduces data redundancy, and enhances operational efficiency.

- Single Sign-On: SSO provides a streamlined user experience by eliminating the need to remember and enter multiple passwords. It also enhances security by reducing the likelihood of poor password practices.

4. Security Enhancements:

- Data Encryption: In the era of increasing cybersecurity threats, strong data encryption is essential to safeguard business-critical information from unauthorized access and breaches.
- Multi-Factor Authentication (MFA): MFA significantly increases account security by adding an extra layer of protection. Even if a password is compromised, an attacker won't be able to gain access without the second authentication factor.

5. Scalability:

- Cloud Technologies: Utilizing cloud technologies allows the application to handle growing data and user load without investing heavily in hardware resources. It provides elasticity, ensuring the application can easily scale up or down based on demand.
- Distributed Systems: Distributed systems improve the reliability and availability of the application. Even if one component fails, others can continue to function, ensuring seamless service delivery.

6. Mobile Compatibility:

- Responsive Design: With the increasing use of mobile devices, having a responsive design ensures users can access the application anytime, anywhere, on any device, enhancing accessibility and user experience.
- Native Mobile Applications: Native mobile applications can take advantage of device capabilities (like notifications, GPS, camera, etc.), providing a richer and more engaging user experience.

7. Automated Testing:

- Tool Implementation: Automated testing can significantly speed up the testing process, improve test coverage, and increase the efficiency and effectiveness of software testing, leading to a more robust application.
- Continuous Integration and Continuous Deployment (CI/CD): CI/CD allows for faster and safer deployments, helps in early identification and resolution of bugs, and ensures a reliable and up-to-date application.

The above justifications underline the importance of each suggested improvement and emphasize their potential impact in enhancing the functionality, usability, and user satisfaction of the BAUHINIA Inventory Control Application.

4.8 Suggestions for Further Development

4.8.1 A. Proposed Additional Features or Changes

1. AI-Powered Predictive Analytics:

One suggested feature is the implementation of AI-powered predictive analytics. These algorithms can analyze historical data to predict future demand, helping to prevent stockouts and overstock situations. Predictive analytics could also aid in forecasting sales, identifying trends, and optimizing inventory levels, all crucial for efficient inventory management.

2. Real-time Notifications and Alerts:

Real-time notifications and alerts could be an excellent addition to the application. This could involve alerts for low inventory levels, sudden changes in demand, or items that are not moving. Timely notifications would allow BAUHINIA to react promptly to any inventory-related issues and maintain optimum stock levels.

3. Integration with E-commerce Platforms:

Given the rise of online retail, integrating the inventory system with e-commerce platforms could provide significant benefits. This could help in tracking online sales in real-time, maintaining accurate inventory levels, and preventing over-selling or under-selling.

4. RFID Technology Integration:

RFID (Radio Frequency Identification) technology could be integrated into the inventory management process. RFID tags can enable real-time tracking of items, from the warehouse to the retail store, and even to the customer. This technology can greatly enhance inventory accuracy and efficiency.

5. Barcode Scanning Mobile App:

Developing a complementary mobile application with barcode scanning capabilities could streamline the process of inventory counting, goods receiving, and order picking. This feature could help reduce human error, improve accuracy, and save time in inventory management.

6. Vendor Management:

A vendor management module could also be incorporated. This could help BAUHINIA to keep track of all vendor details, purchase orders, invoices, and vendor performance. Efficient vendor management could help in negotiating better terms, ensuring timely delivery, and maintaining good vendor relationships.

7. Multi-warehouse Management:

If BAUHINIA operates across multiple warehouses, a multi-warehouse management feature would be beneficial. This feature would allow for tracking inventory across different warehouses, transferring stock between warehouses, and managing warehouse-specific inventories.

8. Customized User Roles and Access:

Introducing customized user roles and access levels can enhance the security of the application. It would ensure that employees can only access and perform tasks that are relevant to their job roles, thereby minimizing the risk of accidental data modifications or breaches.

These suggested features and changes are intended to further enhance the functionality, efficiency, and usefulness of the BAUHINIA Inventory Control Application. While these recommendations are based on standard best practices and trends in inventory management, they should be carefully evaluated and customized to BAUHINIA's specific needs and circumstances.

4.8.2 B. Justification for Each Suggested Development

1. AI-Powered Predictive Analytics:

Implementing AI-powered predictive analytics would allow for more efficient inventory management. By accurately predicting future demand, BAUHINIA can ensure they have the right amount of stock at the right time, reducing storage costs and the risk of stockouts or overstocks. AI can also help in identifying sales trends and making data-driven decisions, which would positively impact business performance.

2. Real-time Notifications and Alerts:

Real-time notifications would enable BAUHINIA to take immediate action in response to sudden changes in inventory. This can lead to enhanced operational efficiency and improved customer satisfaction by ensuring that products are always available when needed.

3. Integration with E-commerce Platforms:

Integration with e-commerce platforms is crucial in today's digital age. It allows for real-time tracking of online sales and inventory management, which is essential to keep up with the fast-paced online retail market. This feature can significantly improve inventory accuracy and customer satisfaction by ensuring that products listed online are actually in stock.

4. RFID Technology Integration:

RFID technology can revolutionize inventory management by allowing for real-time tracking and automatic updating of inventory levels. This would greatly increase inventory accuracy, reduce manual labor, and enhance overall operational efficiency.

5. Barcode Scanning Mobile App:

A mobile application with barcode scanning capabilities would streamline many inventory management processes. It would make tasks like inventory counting, goods receiving, and order picking quicker and more accurate, thereby saving time and reducing the likelihood of errors.

6. Vendor Management:

Efficient vendor management is key to maintaining a well-functioning supply chain. By having all vendor-related information in one place, BAUHINIA can easily manage their relationships with vendors, ensure timely delivery of products, and negotiate better terms.

7. Multi-warehouse Management:

If BAUHINIA operates across multiple warehouses, managing each one separately can be challenging. A multi-warehouse management feature would simplify this process by allowing for centralized tracking and management of all warehouses, thereby improving efficiency and accuracy.

8. Customized User Roles and Access:

Customized user roles and access levels can significantly enhance the security of the application. It ensures that employees only have access to information that is relevant to their job, reducing the risk of accidental or intentional data breaches.

In conclusion, these proposed developments aim to further enhance the BAUHINIA Inventory Control Application, making it more robust, efficient, and useful to BAUHINIA. However, it's important to note that the implementation of these features should align with BAUHINIA's business goals, operational capacity, and budget considerations.

4.9 Conclusion of Critical Review

4.9.1 A. Summary of the Critical Review and Evaluations

This critical review and evaluation have provided an in-depth assessment of the BAUHINIA Inventory Control Application from its design stage to the development, testing, and final performance of the business application. The review process has allowed us to identify not only the strengths and achievements but also the challenges and areas for improvement at each stage of the application lifecycle.

Review of Design Stage:

The design stage played a critical role in defining the overall structure and functionality of the application. The effective use of design tools, such as flowcharts and wireframes, aided in visualizing the application's user interface and workflow, thereby ensuring alignment with the problem definition statement and initial requirements. However, there were challenges related to feature prioritization and usability design, which could be improved upon in future projects.

Review of Development Stage:

The development stage successfully transformed the design into a functional application. Various development tools and techniques were effectively employed to build the application and ensure it met the initial requirements. Despite some issues related to debugging and integration, the overall development process was successful.

Review of Testing Stage:

The testing stage was instrumental in ensuring the quality of the application. The use of various testing tools and techniques, along with a comprehensive test plan, helped uncover and resolve bugs and performance issues. While the testing process was generally

successful, areas for improvement were identified, particularly in terms of increasing test coverage and improving automated testing capabilities.

Review of Business Application:

The BAUHINIA Inventory Control Application proved to be effective in addressing the problem statement and meeting the initial requirements. It demonstrated significant strengths, such as robust inventory management functionalities, a user-friendly interface, and strong integration capabilities. Nonetheless, some weaknesses were also identified, including potential improvements in navigation, data security, and mobile compatibility.

Recap of Initial Risks Identified:

The initial risk identification process was beneficial in foreseeing potential challenges and implementing appropriate mitigation strategies. Most of the risks identified were successfully addressed during development, while others materialized and were effectively managed. The success in risk management could be attributed to thorough planning and proactive risk management strategies.

Opportunities for Improvement and Suggestions for Further Development:

Finally, the review process yielded valuable insights into potential improvements and further development of the BAUHINIA Inventory Control Application. Feedback from peers and users highlighted areas such as enhancing the user interface, incorporating advanced analytics, improving security features, and ensuring scalability. These insights will be invaluable in refining the application in future iterations.

In conclusion, the development of the BAUHINIA Inventory Control Application has been a learning journey that offered numerous insights into the process of designing, developing, and testing a business application. The critical review process has not only highlighted the successes and strengths of the application but also provided a roadmap for future

enhancements and developments. Moving forward, these findings will serve as a valuable guide in refining the BAUHINIA Inventory Control Application and ensuring its continued success in serving the needs of BAUHINIA and its users.

4.9.2 B. Reflective Discussion on the Learning Experience and Overall Development Process

The process of developing the BAUHINIA Inventory Control Application has been an immensely enriching learning journey. This undertaking has underscored the importance of each stage in the software development life cycle, from design and development to testing, as well as the importance of risk management and user feedback.

Design Learning Experience:

In the design stage, we learned that careful planning and foresight play an essential role in setting up a project for success. The use of design tools such as wireframes and flowcharts was particularly enlightening, providing us with a visual representation of the user interface and flow of the application. This experience underscored the importance of iterative design, as we revisited and refined our designs several times based on feedback and new insights.

Development Learning Experience:

During the development phase, we gained practical experience with various development tools and languages. This stage underscored the importance of code organization and readability, as well as the value of effective debugging and problem-solving skills. We learned that while initial coding can be quick, refining and debugging the code can be time-consuming, and thus, it is essential to allocate time and resources accordingly.

Testing Learning Experience:

The testing stage was a significant learning experience. We learned the value of rigorous testing to uncover bugs and issues that might not be apparent during the development stage. We discovered that developing comprehensive test cases and plans is crucial to ensure all aspects of the application are adequately tested. This phase also emphasized the value of automated testing, which we found to be a powerful tool for improving efficiency and reducing human error.

Risk Management Learning Experience:

The process of identifying and managing risks was another valuable learning experience. We learned that anticipating potential problems and proactively addressing them can help mitigate many issues before they even arise. This process also highlighted the importance of flexibility and adaptability, as some risks materialized despite our efforts to mitigate them.

User Feedback and Improvement:

Finally, the importance of user feedback in the development process was underscored during this project. The constructive feedback received during the review process was instrumental in identifying the application's strengths and weaknesses. It highlighted areas for improvement and provided valuable insights that we would not have obtained otherwise. This experience emphasized the need for user-centric design and the value of iterative development, where feedback is continually incorporated to refine and improve the product.

Overall, this project has been a holistic learning experience that has broadened our understanding of software development, from initial planning and design to implementation, testing, and feedback incorporation. The skills and knowledge acquired during this process will be invaluable in our future projects and endeavours. We look forward to applying these learnings in our next steps, as we refine and enhance the BAUHINIA Inventory Control Application.

Conclusion

In Activity 1, I examined the problems of the current inventory system, defined user and system requirements, and outlined potential risks. I then created a detailed Software Design Document (SDD) that included comprehensive system design, coding standards, and risk mitigation strategies.

Activity 2 involved the selection of software development tools and techniques, with a focus on Agile and Scrum methodologies.

In Activity 3, I developed and reviewed the BAUHINIA project. Peer reviews, system requirements alignment, and comprehensive documentation were key components of this activity. I also proposed future improvements, including advanced analytics, AI integration, and user customization.

Finally, Activity 4 provided a critical review of the project, assessing the design, development, testing stages, and overall performance of the business application. This stage also revisited the risk management process and suggested areas for future enhancements. The project offered valuable learnings and established a solid groundwork for the further development of the BAUHINIA Inventory Control Application.

References

Indeed, 2023. What is a problem statement?

Available at: <https://www.indeed.com/career-advice/career-development/what-is-a-problem-statement>

(Accessed: 15 May 2023).

Performance Validation, 2023. User Requirements.

Available at: <https://www.perfval.com/technical-discussion/user-requirements/#:~:text=User%20requirements%20are%20just%20what,which%20product%20should%20be%20made.>

(Accessed: 17 May 2023).

TechTarget, 2023. What are requirements types?

Available at: <https://www.techtarget.com/searchsoftwarequality/answer/What-are-requirements-types>

(Accessed: 19 May 2023).

Sauter, V., 2023. System Requirements.

Available at:

<https://www.umsl.edu/~sauterv/analysis/F2015/System%20Requirements.html#:~:text=System%20requirements%20is%20a%20statement,to%20satisfy%20the%20customer's%20requirements.>

(Accessed: 21 May 2023).

Techopedia, 2023. System Requirements.

Available at: <https://www.techopedia.com/definition/4371/system-requirements>

(Accessed: 24 May 2023).

Gartner, 2023. Risk Identification (RI).

Available at: [https://www.gartner.com/en/information-technology/glossary/risk-identification-ri-#:~:text=Risk%20identification%20\(RI\)%20is%20a,%2C%20damage%2C%20loss%20or%20reputation.](https://www.gartner.com/en/information-technology/glossary/risk-identification-ri-#:~:text=Risk%20identification%20(RI)%20is%20a,%2C%20damage%2C%20loss%20or%20reputation.)

(Accessed: 27 May 2023).

Indeed, 2023. Risk Identification.

Available at: <https://www.indeed.com/career-advice/career-development/risk-identification>

(Accessed: 29 May 2023).

Lucidchart, 2023. How to create software design documents.

Available at: <https://www.lucidchart.com/blog/how-to-create-software-design-documents>

(Accessed: 2 June 2023).

GeeksforGeeks, 2023. Design Documentation in Software Engineering.

Available at: <https://www.geeksforgeeks.org/design-documentation-in-software-engineering/>

(Accessed: 4 June 2023).

TutorialsPoint, 2023. System Analysis and Design Overview.

Available at:

https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_overview.htm#:~:text=System%20analysis%20is%20conducted%20for,what%20the%20system%20should%20do.

(Accessed: 6 June 2023).

Simplilearn, 2023. What is a Feasibility Study in Project Management?

Available at: <https://www.simplilearn.com/feasibility-study-article#:~:text=As%20the%20name%20implies%2C%20a,project%20may%20not%20be%20doable>.

(Accessed: 9 June 2023).

Tatvasoft, 2023. Top 12 Software Development Methodologies and its Advantages & Disadvantages.

Available at: <https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/>

(Accessed: 11 June 2023).

Khan, K., 2023. System Development Methodologies.

Available at: <https://www.slideshare.net/KashifKhan76/system-development-methodologies-64648700>

(Accessed: 14 June 2023).

GeeksforGeeks, 2023. What is System Design? Learn System Design.

Available at: <https://www.geeksforgeeks.org/what-is-system-design-learn-system-design/>

(Accessed: 16 June 2023).

GeeksforGeeks, 2023. Coding Standards and Guidelines.

Available at: <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>

(Accessed: 19 June 2023).

BrowserStack, 2023. What is a Test Plan?

Available at: <https://www.browserstack.com/guide/test-planning#:~:text=a%20Test%20Plan%3F-,What%20is%20a%20Test%20Plan%3F,correctly%20%E2%80%93%20controlled%20by%20test%20managers.>

(Accessed: 21 June 2023).

Guru99, 2023. What is a Test Case? Example.

Available at: <https://www.guru99.com/test-case.html>

(Accessed: 24 June 2023).

RIB CCS, 2023. The Role of Maintenance and Support in Software Implementation.

Available at: <https://ribccs.com/role-of-maintenance-and-support-in-software-implementation/>

(Accessed: 27 June 2023).

The New Stack, 2023. A Comparison of 5 Popular Application Deployment Strategies.

Available at: <https://thenewstack.io/deployment-strategies/>

(Accessed: 29 June 2023).

TechTarget, 2023. Risk Mitigation.

Available at: <https://www.techtarget.com/searchdisasterrecovery/definition/risk-mitigation>

(Accessed: 2 July 2023).

Guru99, 2023. Top 50 Software Development Tools in 2023.

Available at: <https://www.guru99.com/software-development-tools.html>

(Accessed: 5 July 2023).

Software Testing Help, 2023. Top 60 Best Software Development Tools in 2023.

Available at: <https://www.softwaretestinghelp.com/software-development-tools/>

(Accessed: 7 July 2023).

UpTech, 2023. Ultimate Guide to Software Development Methodologies in 2023.

Available at: <https://www.uptech.team/blog/software-development-methodologies>

(Accessed: 10 July 2023).

Wiley, 2023. What is Peer Review?

Available at: <https://authorservices.wiley.com/Reviewers/journal-reviewers/what-is-peer-review/index.html#:~:text=Peer%20review%20is%20designed%20to,invalid%20or%20poor%20quality%20articles.>

(Accessed: 12 July 2023).

BioMed Central, 2023. Peer Review Process.

Available at: <https://www.biomedcentral.com/getpublished/peer-review-process>

(Accessed: 14 July 2023).

Quandary Consulting Group, 2023. Business Application Development: The Key to Growth.

Available at: <https://quandarycg.com/business-application-development-growth/>

(Accessed: 17 July 2023).

Document360, 2023. The Ultimate Guide To Creating A User Manual.

Available at: <https://document360.com/blog/creating-a-user-manual/>

(Accessed: 19 July 2023).

HubSpot, 2023. What is Technical Documentation? A Comprehensive Guide.

Available at: <https://blog.hubspot.com/service/technical-documentation#:~:text=with%20the%20basics.-,What%20is%20technical%20documentation%3F,design%20process%2C%20and%20intended%20applications.>

(Accessed: 15 July 2023).

TechRepublic, 2023. 10 things you can do to improve your application development.

Available at: <https://www.techrepublic.com/article/10-things-you-can-do-to-improve-your-application-development/>

(Accessed: 17 July 2023).

Intelegain, 2023. 30 Best App Ideas for Startups to Launch in 2023.

Available at: <https://www.intellegain.com/app-ideas-for-startups-to-launch-in-2023/>

(Accessed: 19 July 2023).