

Project Title

Stock management system for University -
ActStock

Author

Aminath Muzuna

S2001493

Bsc Computer Science

Table of Contents

Project Title	1
Abstract.....	5
Acknowledgements.....	6
Table of Figures.....	7
Table of Tables	8
1. Introduction	9
1.1 Aims and the Objectives:	9
1.2 Project Scope	10
2. Literature Review	10
2.1 University	11
2.2 Stock Management System in university.....	12
2.3 Using Excel or spreadsheet to manage inventory/stock	13
2.4 Database Management Systems in Inventory Control	13
2.5 Effective strategies to inventory management.....	14
3. Requirements.....	14
3.1.1 Stock value Monitoring.....	15
3.1.2 Adding items	15
3.1.3 Take items out.....	15
3.1.4 Generate a report	16
3.1.5 Home Page	16
3.1.6 Register and login	16
3.2 Analysis of the Requirements:	16
3.2.1 Functional requirements:.....	17
3.2.2 Non-functional requirements:	19
3.2.4 Requirement prioritization using MoSCoW :.....	19
4. Methodology.....	23
4.1 Waterfall Method	23
4.2 Spiral Method.....	23
4.3 Agile Method.....	24
4.4 Summary	24
4.5 Sprint 1:.....	25
4.6 Sprint 2:.....	25
4.7 Sprint 3:.....	25

4.8	Sprint 4:.....	25
5.	Design.....	26
5.1	Software Architecture.....	26
5.2	Use Case Diagram	27
5.3	Class Diagram.....	28
5.4	State Diagram.....	29
5.5	User –Interface.....	32
6.	Implementation	35
6.1	Programming Language	35
6.1.1	Java.....	35
6.1.2	Maven	36
6.2	DB SQLite.....	36
6.3	IntelliJ IDEA	36
6.4	Sprints	37
6.4.1	Sprint 1:.....	37
6.4.2	Sprint 2:.....	37
6.4.3	Sprint 3	37
6.4.4	Sprint 4:.....	38
6.5	Implementation Overview:	38
6.6	Database of the system:	39
6.6.5	Database connection method:.....	40
6.6.6	Register method:.....	41
6.6.7	Login method:	41
6.6.9	User Interface of the Register and Login page:	42
6.6.10	Inventory Controlling method:	43
6.6.11	Monitor controller method:	44
6.6.12	Report Controller Method:	45
6.6.13	Update Controller method.....	46
6.6.14	Controller of the View switching and User Management:	47
6.6.15	JavaFX Application Manager	48
6.7	Testing:.....	48
6.7.1	Sprint 1:.....	48
6.7.2	Sprint 2:	49
6.7.3	Sprint 3:.....	50
6.7.4	Sprint 4:.....	50
7.	Project evaluation	51

7.1 Choice of technologies	51
7.2 Development of the system:.....	51
7.3 Performance and Limitations of the system	52
7.4 Improvements and Expandability of the system	52
7.5 The learning Outcomes	52
7.6 My Self Reflection	53
8. Conclusion	54
References/Bibliography.....	55

Abstract

In this era of digitalization, a lot of universities including the Islamic university of Maldives (IUM) face challenges in managing their stock or inventory system. This most of the times results in lack of transparency and insufficient usage of the stock items. To address these issues, mostly the issues faced in IUM, a University Stock Management System named ActStock was designed and implemented. ActStock provides a centralized platform for the stock management, streamlining the procurement process and generating the reports for a better decision making. This also provides a user friendly interface for easy access and efficient management of stock across the campus. Despite being able to meet all the functional requirements, there are still ample for future improvements and more functionalities in order to enhance the effectiveness of the ActStock.

Acknowledgements

I would like to extend my heartfelt and sincere gratitude and thanks to my supervisor Mrs.Udhuma Abdul Latheef, for her endless support through out this 3 year course of Computer Science. Thank you , miss for always been very helpful and responsive to our questions and doubts even though most of them were very silly questions.

Past few months was a very hard time for me. I lost the first man I have ever loved, my beloved father, just few months back. I fell a lot behind in my studies and life too, due to the huge loss. But my classmates and family helped me to get back on track. Therefore, thank you all for every little thing you guys did for me.

Last but not the least, to the most supportive mom anyone will ever get, who was the reason I am able to be a strong woman and have a good education in a high prestige college, I thank you for all the things you did for me though out my degree. And my father whom I love dearly, thank you for motivating me to finish my degree always. May Allah Almighty bless your soul with Jannah.

Table of Figures

Figure 1Figure 1 . Students per university (Tight,2011))	12
Figure 2Waterfall Model (Sherrell, 2013)	23
Figure 3. Spiral method (Doshi, Jain and Gala, 2021)	24
Figure 4The MVC Architecture (Damilola, 2023)	27
Figure 5UseCase diagram of ActStock	28
Figure 6Class diagram for ActStock	29
Figure 7State Diagram	29
Figure 8MVC pattern in code	38
Figure 9 SQL Item	39
Figure 10SQL usage.....	39
Figure 11SQL User.....	40
Figure 12SQL sqlite_sequence.....	40
Figure 13. database connection	40
Figure 14. registration method	41
Figure 15. login method	41
Figure 16. Error message for attempting to login with wrong data	42
Figure 17User Interface of the Register and Login page:	43
Figure 18Inventory Controlling method	44
Figure 19Monitor Controlling Method	45
Figure 20 Report Controller method	46
Figure 21Update Controller method	47
Figure 22Controller of the View switching and User Management	48
Figure 23Manager.....	48

Table of Tables

Table 1: Universities by national region(Tight,2011)	11
Table 2Proportions of population, students and universities by national region (Tight, 2011).....	12
Table 3. requirements Homepage.....	20
Table 4. Inventory retriever requirement	20
Table 5Monitor Controller requirements	21
Table 6Report controller requirements.....	21
Table 7Update controller requirements.....	21
Table 8Login and register requirements.....	21
Table 9Integration requirements	22
Table 10 Class descriptions	32
Table 11. Homepage UI.....	34
Table 12 Login and Register test	49
Table 13Database test	49
Table 14HomePage test	49
Table 15Update controller test	49
Table 16Inventory Controller test	50
Table 17Monitor Controller test	50
Table 18Report Controller test 1.....	50
Table 19Report Controller test2.....	50

1. Introduction

Imagine a world where stock management in all the universities and academic institutions are made effortless. Imagine a scenario where stock monitoring and generating reports with the popularity percentage of a particular stock item are seamlessly integrated into a single system in all the universities, including the universities of Maldives. In the world of higher education, where efficiency and organization are really needed, the necessity for a Stock Management System becomes evident. Despite technological advancements, most of the universities such as the Islamic University of Maldives (IUM) still face challenges in managing their stock, resulting in wasted stock items and operational inefficiencies. In this situation, a comprehensive Stock Management System made particularly for university emerges as a great solution to streamline stock-related processes, improve the control of inventory and enhance overall operational effectiveness of the university. By harnessing the technology's power, universities like Islamic University of Maldives can optimize their stock management practices by ensuring a smooth and efficient flow of the stock items within the institution.

1.1 Aims and the Objectives:

The aim of this project is to create a stock management system (in this iteration I will focus on stationeries) for universities, and I have particularly based this system on the issues and needs of Islamic University of Maldives's staff while managing their stationery stock. This system will be a desktop based GUI Application that will allow the stock managing staff to easily keep a track of the stock items while introducing necessary and new stationery items into the stock according to their usage and popularity.

The objective is to perform enough research into the stationery stock management system in the universities, especially in Islamic University of Maldives. And to have an understanding of similar applications, their benefits and limitations and the functionalities these systems provides.

And when adequate amount of research has been conducted with regard to stationery stock management systems in the university, the research findings will be utilized in order to plan the project and then establish the overall requirements of the stock management system.

1.2 Project Scope

Following the background research into why a stationery stock management system is required in universities especially in the Islamic University of Maldives, this section would give an outline on the project scope with regard to the system objectives and the project management and the prototyping decisions that were made before the development process of the system.

The platform called as “ActStock” would be built as a desktop GUI application. This is built aiming to support any OS operating system.

1.3 Structure of the report

Chapter 2 of this report would mostly focus on the literature review and other research which will tackle the stock management system of the university. It will also analyse the strengths and weaknesses of various software that are used to manage the stock in a university. Also, there will be a review on the technologies that will be used in order to implement this stationery stock management system – ActStock. Chapter 3 will be focusing on the application requirement which will include all functional and non-functional requirements of the system ActStock. Chapter 4 is about the methodology that is applied to develop this project. Chapter 5 contains all the designs and other necessary information that are needed in order to build this project. In chapter 6 it will focus on the implementation of the project. The next chapter 7 will have an overall evaluation and self-reflection on this project. This will also include any future improvements if needed any. And at last there will be a conclusion for this project which summarizes the whole document in few lines.

2. Literature Review

2.1 University

There are over 150 higher education institutions or universities in the United Kingdom. (Tight, 2011). This is the case for any part of the world as well. The study also shows that from the population of residents in a nation, most of the population percentage is of students. This means there will be a lot of staff working in the university industry in these nations. And in another recent study it is said that from 2005 to 2015, the number of academic staff who are working in Canada as part-time increased by 79%, while the number of tenure-stream faculty increased by 14% and the number of students increased by 28%. (Toni, 2018). And since the higher education popularity across the world is increasing day by day, the university faculties and staff percentages are expected to grow continuously.

A university is a place where a wide range of academic degrees and courses are offered. In a university students get to learn, do research and grow intellectually every day. Universities consists of several department or sections and various faculties. A lot of academic and administrative staff work in the functioning of these faculties and departments. And for the university to function well, these staff use a lot of different items or stationeries every day.

National Region	1	2	3	4	5	Total
England	89	19	3	10	13	133
Scotland	14		2			161
Wales	3	8				10
Northern Ireland	2				1	3
United Kingdom	108	27	5	10	14	162

Table 1: Universities by national region(Tight,2011)

				Universities from Table 1	
National region (%)	Population (%)	Students (%)	1	1+2+3	1+2+3+4+5
England	51,809,700 (84)	2,052,380(83)	89 (82)	110(80)	133(82)
Scotland	5,194,000(8)	215,635(9)	14(13)	16(12)	16(10)
Wales	2,993,300(5)	148,930(6)	3(3)	10(7)	10(6)

Northern Ireland	1,788,900(3)	48,240(2)	2(2)	2(1)	3(2)
United Kingdom	61,792,000(100)	2,465,185(100)	108(100)	138(100)	162(100)

Table 2 Proportions of population, students and universities by national region (Tight, 2011)

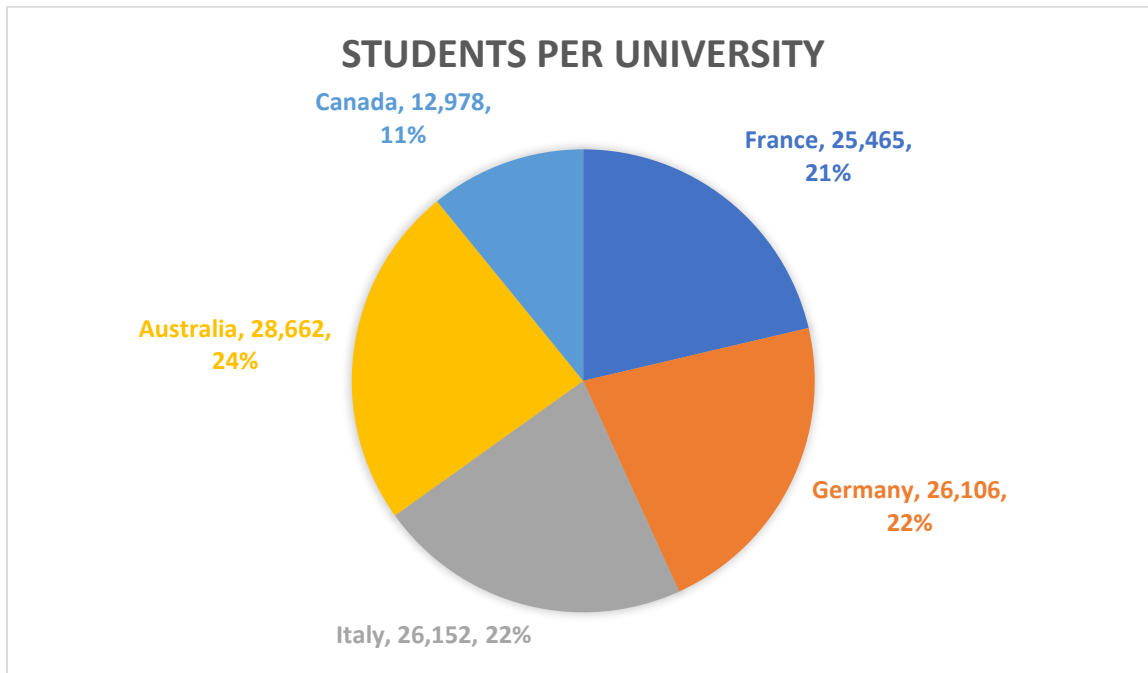


Figure 1 Figure 1 . Students per university (Tight, 2011)

2.2 Stock Management System in university

According to studies , a stock management system is a necessity of every university. For the services to occur continuously, the universities need continuous supplies of items or stationeries to it's shareholders or staff. A stock management system makes it easier for the procurement and stock managing staff to keep a track of the items in the university. Stock managing staff would know the type of items that are in the stock using the system. This can help the stock managing staff to check the market and introduce several new items into the stock. Also, the procurement team should not have to worry for under-stocking and over-stocking of the items present in the university. This will aid the procurement team in doing the procurement process in the right time. For example, if the item 1 is getting low, the team can start the procurement process as soon as possible as the finance process also gets involved while buying the items for the university's stock. And the finance process for the universities most of the time take a lot of time. By initializing the procurement process sooner when the item is low, university staff will not have to face any difficulty in getting the necessary items

whenever they need in order to provide a commendable service for their students. Unlike the traditional way of stock management which is the manual inventory management technique, having a stock management system will save the stock managing staff and the procurement team a lot of time. This is because in the manual inventory management technique will have to count and manually write in books or papers the record of each and every type of item in the stock. Also, using a stock management system instead of the traditional way of entering the numbers manually will save a lot of energy for the stock managing staff and procurement staff as well, as they need to learn a lot of mathematical skills and formulas if they want to record and keep the track of inventory using the manual inventory management technique. And doing manual calculations for a lot of items in the stock might cause errors to occur, as humans are very likely to make mistakes especially when we do complex works.

2.3 Using Excel or spreadsheet to manage inventory/stock

Spreadsheet or Microsoft Excel are the most frequently used platform for inventory management and analysis. Microsoft Excel could perform statistical tests and regression analysis with the aid of add-in software like What-If Solver and @Risk. But, the general purpose spreadsheet software is not designed for the tasks that are more complex. Therefore, too much time and energy may be wasted in the spreadsheet skills learning. (Jiang, 2010). This means businesses or institutions like universities need a stock management system or desktop application instead of the spread sheets or Excel Sheets. Introducing a desktop application to universities who use Excel sheets and spread sheets, will save the university staff who manages the stationery stock and the procurement staff's precious time.

2.4 Database Management Systems in Inventory Control

In the article "Blood Bank Management and Inventory Control Database Management System" by Shah et al. (2022), the authors propose a database management system for inventor control for blood bank. The system utilizes a barcode scanner to track blood products and includes features like inventory alerts and expiration date tracking. The author argues that the system could improve the accuracy of inventory and reduce waste in the operations of blood bank. This can be applied in the stock management system of university. The levels of item can be made seen and this can be used to

determine the Alert levels such as “High” , “Low” and “Medium”. Applying this feature will help the university in maintaining the items without going under or over stock.

2.5 Effective strategies to inventory management

The study article "Inventory Management in Mass Customization Operations: A Review" by Guo et al. (2019) shows a useful review of inventory management practices in mass customization operations, that could be applied to the unique necessities of universities. The authors stress the importance of balancing the levels of inventory with customization requirements that would ensure that the necessary stationery items are available for the students and staff while avoiding the overstocking of stationeries. This can be applied and can be very helpful in the university in order to manage and control it's stock items. This would make the university continue it's services and operations effectively. All the items could be introduced into the stock according to it's usability and demand.

3. Requirements

According to Maguire and Bevan (2002) understanding the user requirements is an integral part of the design of information systems as well as is critical to the success of any user interactive system. As stated before, to capture the system's user requirements I conducted a survey among 50 staff working in a university. They were asked to fill a google form with various questions about the stock management and its functionalities. Several different answers are obtained and using these answers I selected the key functionalities of the system.

.Functional requirements include inventory management capabilities like stock tracking, restocking and notifications. It also includes user roles and permissions. Additionally, generation for items, barcode scanning, reporting, analytics and integration with existing systems such as procurement and accounting are also key requirements for the system.

3.1.1 Stock value Monitoring

The staff who took the survey questions said that they wanted to know the count of each and every item in the stock. Also they wanted to know if the number of a specific item in the stock is low or high.

Therefore, the ActStock – stock management system must be able to show the count of the number of items in the stock after adding or removing from the stock. And also it should be able to detect if the stock of a particular item is lowering in number and Alert the staff using the system. The items with low values must show the Alert column as High, as the staff should be concerned about that particular item more and do the procurement process to order and get those items into the stock as soon as possible.

3.1.2 Adding items

As the participants of the survey wanted to be able to add items into the stock and show the final values after adding the items into the stock.

Therefore, the ActStock – stock management system must have a function where users should be able to add new items into the stock and a function where the user should be able to add more to an existing item of the stock. Finally, it will show the total amount of each and every item in the stock, after the adding function. This function should alter the Alert level of the item accordingly, for example an item which was with High alert when a number of that particular item is added into the stock, the Alert level might change to Medium or Low accordingly.

3.1.3 Take items out

The survey participants also said that they wanted to be able to take items out of the stock or remove items from the stock.

So, ActStock – stock management system must have the function where the user should be able to take a number of a particular item(s) out of the stock. This should show a change in the total number of that particular item in the stock as the number should be decreasing as the items got removed. This function also should alter the Alert level of the item accordingly. As an example: An item with Medium Alert might change to High after the item removing function occurred.

3.1.4 Generate a report

Participated staff also said that they wanted the stock management system to have a function where they can generate and download reports of the stock, whenever they want.

So, the ActStock- stock management system should use the number of items in the stock and generate a report with the name of each and every item in the stock, the total number of that particular item in the stock, usage percentage of the particular item in the stock and the popularity percentage according to the usage of the item. The generated report should be downloaded in CSV format.

3.1.5 Home Page

Like every other management system, survey participants wanted the ActStock as a stock management system to have a Home Page with all the choice buttons to run the functions of the system.

3.1.6 Register and login

During the survey, the staff wanted to have their own user IDs in the system. So, ActStock – stock management system should have a function where a user should be able to register by entering relevant information and a password. And then with the correct password and information given to the registration, the user should be able to log in to the system. If wrong information or password, there should be no access given. And most importantly, the user should be asked to enter the password twice in order to confirm if he/she gave the correct password for registration.

3.2 Analysis of the Requirements:

Analysis of the requirement is a very significant part of the software development.

This makes sure that the functionality of the system is completely described by the system developers. The requirement of any software is divided into:

- Functional requirement
- Non-functional requirements

Functional requirements show the services that the system must provide, how the system must behave in particular situations and respond to particular inputs. Moreover, functional requirements might also entail what the system must not do. On the other hand, Non-Functional requirements are known as the constraints of the system's services or functions. They can either be applied to the entire system or to particular services or features. (Sommerville, 2011).

3.2.1 Functional requirements:

1. Home page:

- -The system shall show a Homepage with 4 main functions (Add stock, Checkout, Monitor and Report)
- -The system shall also show in which username user is signed in

2. Inventory Controller:

- The system shall allow users to take out particular items from the stock
- The system shall then update the number of items accordingly after removing of item

3. Monitor Controller:

- The system shall show the Alert for the specific items which are with low number of items in the stock.
- Alert shall be showing according to how low the number of that particular item is in the stock (Eg: Item1 with lowest number of items will have the

Alert as “High” shown as the user need to focus more on adding Item1 into the stock as soon as possible.)

4. Report Controller:

- A report shall be generated by the system with item name, usage of the item, number of the particular item left in the stock and the item’s popularity in %
- The report shall be able to download from the system as a CSV file.

5. Update Controller:

- The system shall add new items into the stock by choosing “Add New” function
- The system shall be able to add items which are existing in the stock as well.

6. Login and Register

- It’s the initial page seen when executed
- The system shall ask to register by adding user information and password
- The system shall let the user access the system if the username and password is found in the database. If not, it will show error.

7. Integration of the system:

- The system shall be able to integrate with systems such as finance system to enable data sharing, automated stock replenishment, and accurate financial reporting.

8. Barcode or QR Scanning:

- The system shall scan barcode or QR code in order to identify the items of stock for the inventory updates, and tracking

3.2.2 Non-functional requirements:

1. A User Friendly Interface:
 - The system shall have a user-friendly interface which is designed so that it will be easy for the user to navigate and understand the system
 - The system shall have an intuitive design and colour contrasts which can be supportive to users with visual impairments.
2. Support for any assistive technologies:
 - The system shall be able to be compatible with assistive technologies for support, which are mostly used by disable people. For example: voice recognition and screen reader.
3. The Performance of the system:
 - The system shall be able to handle a huge number of stock items efficiently
4. Response time:
 - The system shall be able to give a fast response and less delays in inventory updates.
5. Auditability:
 - The system shall keep an audit trail of the activities that are stock-related which includes recording who performed specific actions and when they performed that activity.
6. Compliance
 - The system shall comply with any regulations or standards that are applicable and are related to stock management, protection of data, and reporting regarding finance.
7. Accuracy of Data
 - The system shall maintain an accurate and live stock data. It shall minimize the errors in the records of inventory, calculations, and stock transactions.

3.2.4 Requirement prioritization using MoSCoW :

- Must-Have (M) : These are the requirements that are necessary and that must be included in the project for it to be successful

- Should-Have (S) These are the requirements that are important but not as important as the Must-Haves of the project. These tasks can be included if the user have time for that.
- Could-Have (C) These are the requirements that are nice to have but not necessary for a successful project.
- Wont-Have (W) These are the requirements that are low priority and can be included in the future iterations of the project.

According to Ali Khan et al.,(2015) in MoSCoW the two 'o's have no meaning to that like all the other letters, it is used to make a pronounceable acronym.

The below table are the sets of MoSCoW task prioritization for the requirements that I mentioned in this chapter previously

Home Page	
Task	Priority
The system shall show a Homepage with 4 main functions (Add stock, Checkout, Monitor and Report)	MUST
The system shall also show in which username user is signed in	COULD

Table 3. requirements Homepage

Inventory Controller	
Task	Priority
The system shall allow users to take out particular items from the stock	MUST
The system shall then update the number of items accordingly after removing of item	MUST

Table 4. Inventory retriever requirement

Monitor Controller	
Task	Priority
The system shall show the Alert for the specific items which are with low number of items in the stock.	MUST

Alert shall be showing according to how low the number of that particular item is in the stock (Eg: Item1 with lowest number of items will have the Alert as “High” shown as the user need to focus more on adding Item1 into the stock as soon as possible.)	MUST
--	------

Table 5Monitor Controller requirements

Report Controller	
Task	Priority
A report shall be generated by the system with item name, usage of the item, number of the particular item left in the stock and the item’s popularity in %	MUST
The report shall be able to download from the system as a CSV file	COULD

Table 6Report controller requirements

Update Controller	
Task	Priority
The system shall add new items into the stock by choosing “Add New” function	MUST
The system shall be able to add items which are existing in the stock as well.	MUST

Table 7Update controller requirements

Login and Register	
Task	Priority
It shall be the initial page seen when executed	MUST
The system shall ask to register by adding user information and password	MUST
The system shall ask to login by entering user information and password	MUST
The system shall let the user access the system if the username and password is found in the database. If not, it will show error.	MUST

Table 8Login and register requirements

Integration of the system	
Task	Priority
The system shall be able to integrate with systems such as finance system to enable data sharing, automated stock replenishment, and accurate financial reporting.	COULD

*Table 9*Integration requirements

Barcode or QR Scanning	
Task	Priority
The system shall scan barcode or QR code in order to identify the items of stock for the inventory updates, and tracking	COULD

Table Barcode and QR scan requirements

4. Methodology

In the process of the software development, various methodologies are utilized by the developer in order to guide themselves for the processes in the software development lifecycle. Some often used methodologies are; Agile, Waterfall and Spiral methodology.

4.1 Waterfall Method

Waterfall method is one of the most often utilized among the developers. It is also the methodology that utilized among the developers.

Waterfall model is same as following a recipe step-by-step. Each and every phase of the development, such as obtaining requirements of the system, designing the system, coding the system, deploying and maintaining the system, occurs in a strict order. The developer cannot be allowed to move to the next phase if the previous one is not finished. Also, it does not allow for many changes once a phase is accomplished.

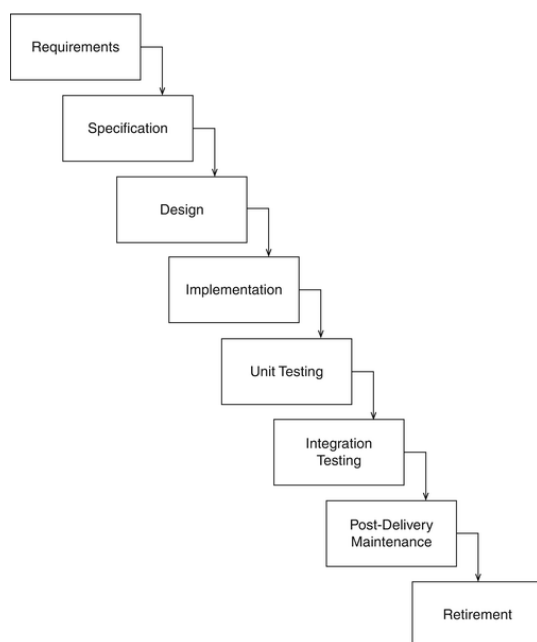


Figure 2 Waterfall Model (Sherrell, 2013)

4.2 Spiral Method

The spiral Model is like a continuous cycle of learning and improving. This method joins the best parts of the iterative development and the waterfall model. A cycle with planning, risk analysing, software development and testing of the software are included in the series of cycles that goes through in the

development process using the Spiral Method. This is like a spiral stair case as each cycle builds upon the previous cycle as you go through it.

The different and special thing about the spiral model is that it allows for adjustments and flexibility. In each cycle, the team learns from their previous work and they receive the feedback of the work done. The feedback helps the team to make improvements and for risk management. This is like taking steps that are small forward, progress evaluation and making changes based on what you learn.

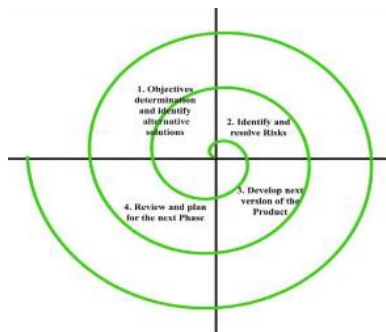


Figure 3. Spiral method (Doshi, Jain and Gala, 2021)

4.3 Agile Method

This is the methodology which follows an iterative approach while developing the system. In this method, all iterations will not have the same length or same process every time. In every iteration, a new release of the software is created and released to the client. Most often, the iterations take two or three weeks. (Sommerville, 2016). As each and every iteration must be published, Agile methodology allows us for testing of each iteration. In this Agile development, iterations are known as sprints. Each sprint is established with their own requirement sets. And these requirements are then implemented and tested in the sprint.

4.4 Summary

In this project of stock management system –ActStock, I used Agile methodology as it offers to accomplish each and every requirement from the Functional requirements in the previous chapter: “Requirements”

And also, as the time limitations and the ability for accepting the new requirements which comes in a later stage of the software development.

Below are the sprints that was applied for my project of stock management system – ActStock:

4.5 Sprint 1:

- In this sprint requirements are gathered and user stories are created. And the system access login and registration function implemented.

4.6 Sprint 2:

- In this sprint the core functionalities of the system gets developed in this sprint.

4.7 Sprint 3:

- This sprint is about UI refinement and Testing process where we are going to improve the user interface (UI) of the system as well as adjusting and adding other functionalities which the staff might find helpful after testing the first sprint.

4.8 Sprint 4:

- This sprint is about focusing on bug fixes, documentation and enhancements of the project. This sprint will identify bugs and issues found during testing.

5. Design

In the project's lifecycle, I followed an approach that is Object-Oriented design approach in order to develop the stock management system –ActStock. The system's interaction and components will be shown in the form of various diagrams below.

At first, I utilized Use-Case diagrams to know the system's functionality from the user's perspective. These diagrams aided me to understand the various actions the user can perform in this system and how they interact with the system.

Then, I created the Class diagrams in order to show the structure of the system and the relationship between the classes. This diagram presented the attributes, methods and associations of every single class and provided a clear view of the architecture of the system.

Eventually, I specified the user interface of the system by outlining how it would appear and function for the users of the system. This process involved determining the layout, components (buttons, text field, and menus) and the interactions between the interface and user.

By applying these techniques of design, I made sure that a systematic and organized approach in developing the stock management system.

5.1 Software Architecture

According to Sharp (1998) Model - View - Controller (MVC) is a pattern which was developed for the graphical user-interface based systems. The software architecture of this stock management system will be based on the Model-View-Controller (MVC).

Below are the three types of classes that are in Model - View - Controller MVC :

1. Model is a class that is utilized to implement data domain logic. These classes are utilized in order to retrieve, insert or update data into the database that is associated with the application. (Lumba and Waworuntu, 2022)

2. View is utilized to create the application's interface. The user interacts with the application using the interface design. (Lumba and Waworuntu, 2022)
3. A controller is a class that is utilized to respond to user requests and then relate them to the model. (Lumba and Waworuntu, 2022)

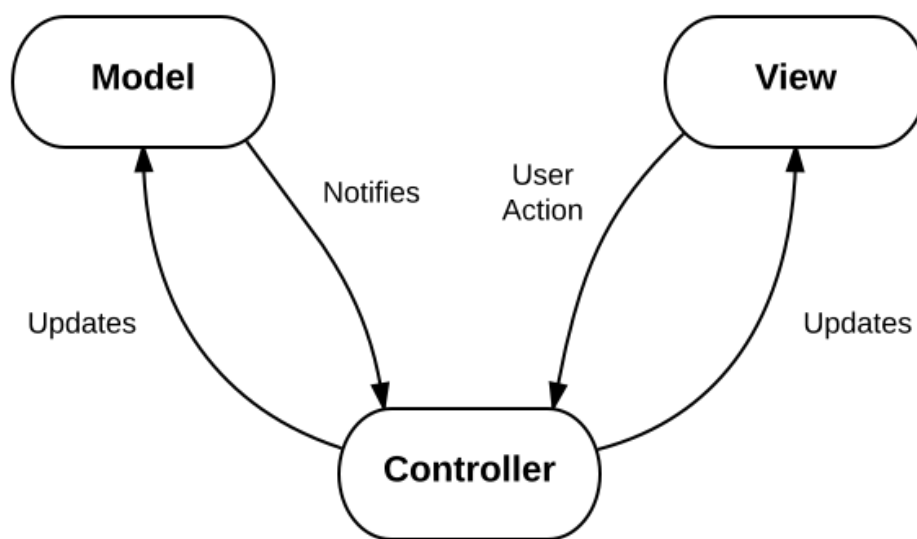


Figure 4The MVC Architecture (Damilola, 2023)

5.2 Use Case Diagram

Use Case Diagrams are very useful:

- to identify the various people or entities that is involved in the system (actors)
- to understand how these actors interact with the system (use cases)

These diagram's aid in designing the system software by showing the several different ways users will use the system. Moreover, the Unified Modeling Language (UML) gives a set of relationships which organize actors and the use cases which allows us to depict the interactions of the user with the system

in a structured and clear manner, which can be used to design the system according to the plan. (von der Maßen & Lichter, 2002).

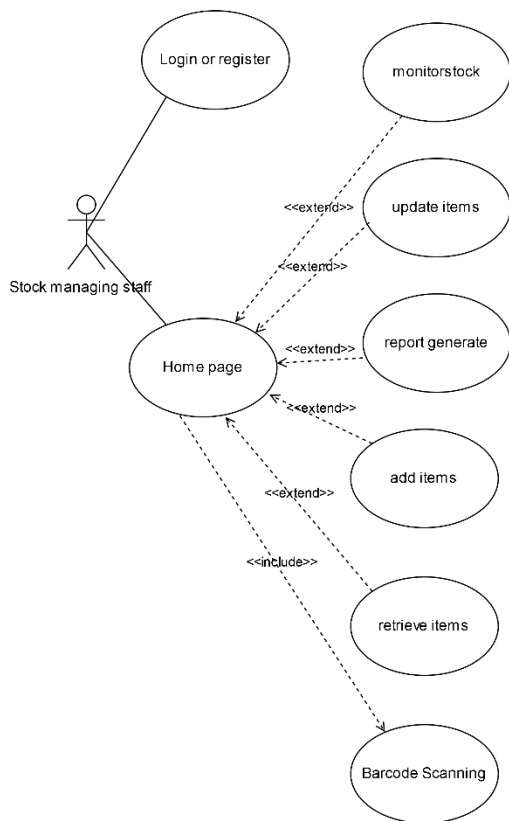
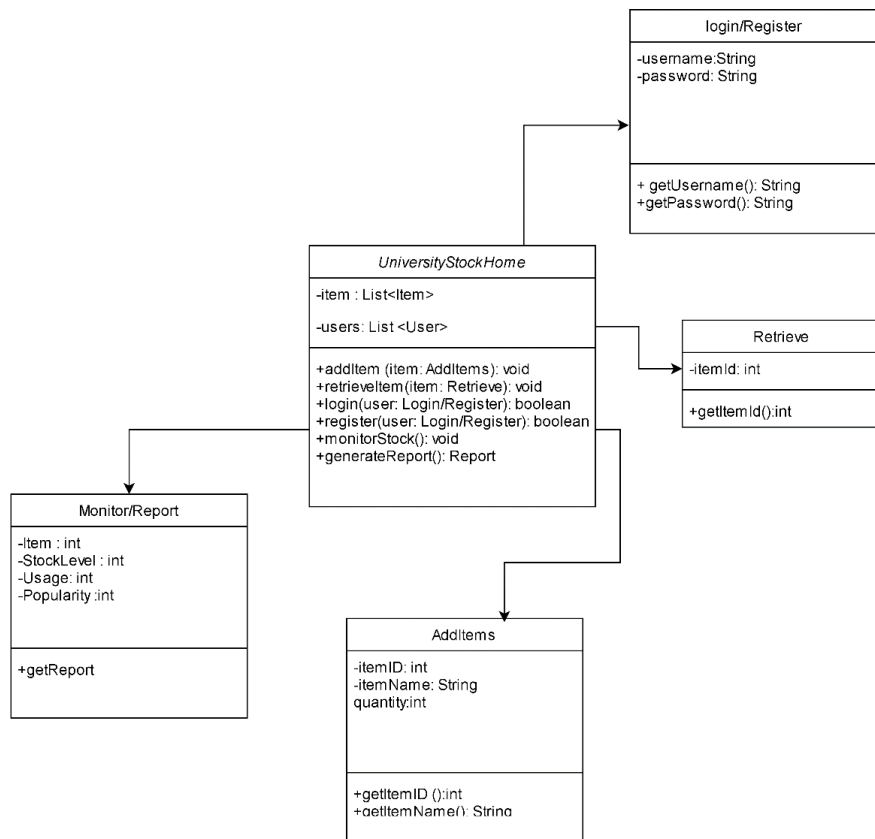


Figure 5 UseCase diagram of ActStock

5.3 Class Diagram

According to Sun and Wong (2005) the UML class diagrams are very helpful while understanding the software system's structure. UML class diagrams describes the classes that are present in the system and how they are related to each other. Simply, a class is a basic building block of the system which shows a concept. Class diagram has characteristics known as attributes and behaviours known as methods associated with it. It will be easy to visualize as well as understand the structure of the system and how several classes interact with each other, by using the class diagram.



S

Figure 6 Class diagram for ActStock

5.4 State Diagram

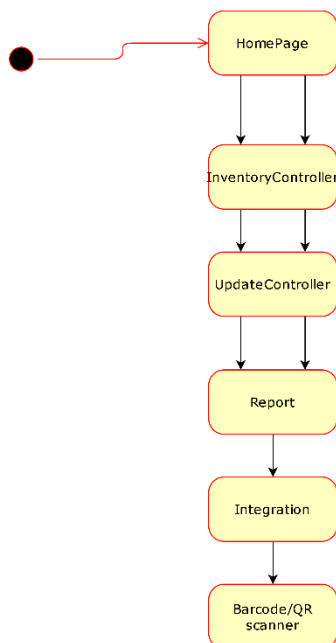


Figure 7 State Diagram

Class	Description
Inventory Controller	<ul style="list-style-type: none"> ○ This class enables to handle the user interfaces and control the displayed information on the screen. ○ This class makes sure that the user enters the necessary information in the correct format. ○ Connects with the database and performs functions like updating and retrieving data from the database. ○ This class also keeps a record of items when they are taken out of the stock which includes details of the user, item and quantity ○ This class displays messages and the content gets updates on the screen in order to notify the user about certain actions..
Monitor Controller	<ul style="list-style-type: none"> ○ This class shows the user's username on the top of the screen ○ This class enables to set up a table in order to display alert data ○ This class retrieves data from the database and it fills the table with the alert information for the items that have a count below a certain threshold. ○ This class handles user interface transitions by allowing the user to switch between the various views like the main view and the login view.
Report Controller	<ul style="list-style-type: none"> ○ This class also shows the username on top of the screen ○ This class sets a table in order to show the report data ○ This class enables the system to retrieve data from the database and then fills the table with the information of items that's needed for the report. This includes stock level, usage and popularity. ○ This class allows the user to switch between various view such as the main and login view. ○ This class enables the system to generate and download reports in a CSV file.
Update Controller	<ul style="list-style-type: none"> ○ This class allows the user to switch between several different views like the main view and the login view ○ This class provides a function which adds new items to the system by inserting the data into a database ○ This class allows the user to update the existing item's count in the system..

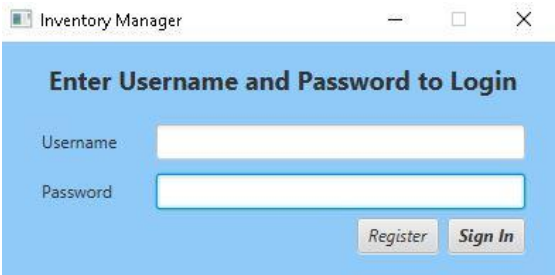
	<ul style="list-style-type: none"> ○ This class also validate the user input in order to ensure that the needed fields are nit empty and the count is a valid number. ○ This class enables the system to retrieve a list with item names from the database connected to it in order to populate a dropdown menu. ○ This class also enables to retrieve the current count of an item from the connected database.
Login Controller	<ul style="list-style-type: none"> ○ This class allows the user to switch between various views such as the login and main view. ○ This class validates the entered username and password by checking with the database connected, in order to determine if the user attempting to login is authenticated or not. ○ This class enables users to register by giving their username, password, full name and role. This checks for any duplicate usernames and also ensures that the entered password match before registering the user. ○ This class enables to perform database operations like querying the user information and inserting new user information into the database ○ This class interacts with the user interface elements such as buttons and text fields in order to get the user input and display alert messages.
Controller	<ul style="list-style-type: none"> ○ This class enables the user to switch between various views such as the main view, inventory view, update, monitor view, report view and login view. ○ This class receives user information and sets it in the user interface in order to display the username on the screen ○ This class loads and initialized the other controllers which are joined with various views, passing the user information to them ○ This class manages the stages and scenes for the display of various views which includes setting the stage properties and scene creation. ○ This class interacts with all the user interface elements like the labels and buttons in order to get the user input and then the events will be triggered accordingly for the view transition.

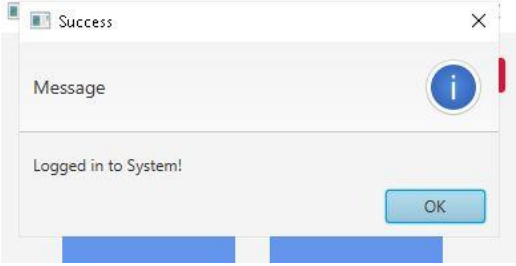
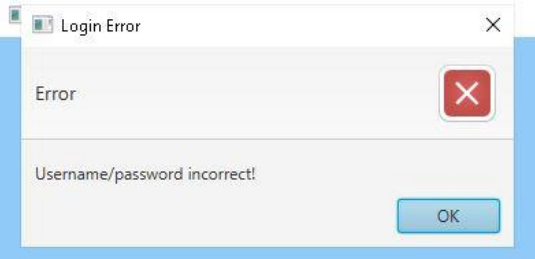
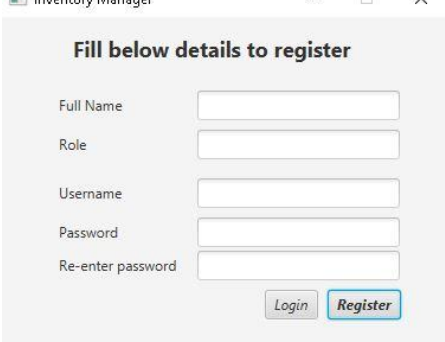
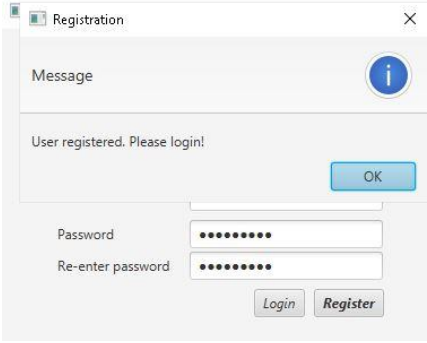
Manager	<ul style="list-style-type: none"> ○ This class functions as the entry point for the application. ○ This class loads the login view by utilizing the JavaFX 'FXMLLoader' and then sets it as the scene for the stage.
---------	---

Table 10 Class descriptions

5.5 User –Interface

User-Interface comes under the field of human-computer interactions. This is the study, panning and designing of how the computer users work with it in order to satisfy their necessities effectively. As stated by Galitz (2007), the User interface (UI) is the most significant part of the system. The reason for this is that to most of the users it is the system. And the goals of the user interface of a system are simple, which is to ensure that the user's experience of using the application or system is easy and exciting or enjoyable.

Login controller	
Screen	Description
	Users get provided a button with the choice sign in or a choice to Register if they do not have an account created yet.

 <p>A 'Success' dialog box with a blue information icon. The message reads 'Logged in to System!' and has an 'OK' button.</p>  <p>A 'Login Error' dialog box with a red error icon. The message reads 'Username/password incorrect!' and has an 'OK' button.</p>	<p>Sign in:</p> <p>The user would need to enter their username and password and then click 'Sign in' in order to proceed. But, if the user forgot the password, they cannot get access to the system yet. The forgot password will be implemented in a later iteration.</p>
 <p>The 'Inventory Manager' window contains a registration form titled 'Fill below details to register'. It includes input fields for Full Name, Role, Username, Password, and Re-enter password. There are 'Login' and 'Register' buttons.</p>  <p>A 'Registration' dialog box with a blue information icon. The message reads 'User registered. Please login!'. Below the message are password fields for 'Password' and 'Re-enter password' (both masked with dots), and 'Login' and 'Register' buttons.</p>	<p>Register:</p> <p>The system user has to enter the necessary information in the fields that are displayed on the screen in order to register an account in the system. After filling the information, when clicked 'Register' the user attempting to login will be given access to the application or system.</p>

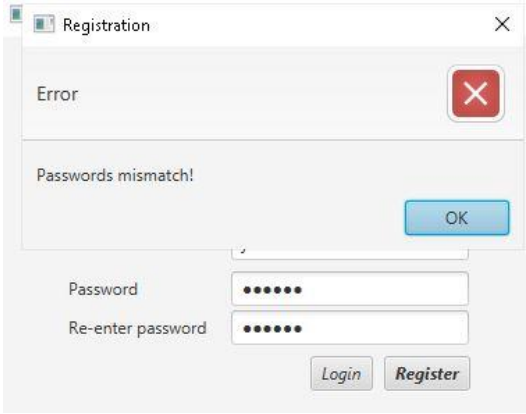
	<p>Error Message:</p> <p>If the user encounters a problem with authentication , the screen will display an error message.</p>
---	---

Table 4: Login/Create Account UI

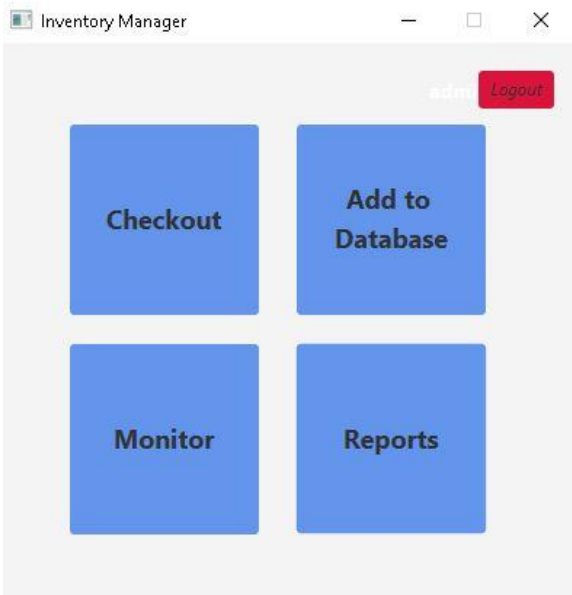
Home Page	
Screen	Description
	<p>In all the views of the system, these will be displayed on top of the screen:</p> <ul style="list-style-type: none"> • The username of the user • A logout button • A back button <p>The user's Home page will be consisting of the following elements or functions:</p> <ul style="list-style-type: none"> • Inventory controller button • Report controller button • Monitor controller button • Inventory controller button

Table 11. Homepage UI

6. Implementation

6.1 Programming Language

In this part of the Implementation chapter, I will search for the choices of a programming language that could be appropriate to carry out this project of stock management system – ActStock. Model View Controller architecture in order to split the software into the main 3 components. They are:

6.1.1 Java

In the process of implementation process of this stock management system – ActStock, Java is selected as the primary programming language. This is because Java has the suitability for the implementation of Model – View – Controller (MVC) architectural pattern as well as its strong support for the Object – Oriented (OO) programming.

Java is very compatible with the software architectural pattern MVC and this made it an ideal selection for the development of the stock management system – ActStock. As the MVC architectural pattern promotes dividing the system into three main components, they are the Model View and Controller. Therefore, Java's features such as the interfaces, classes and inheritance made it possible for me to easily determine the Model, that represents the data and business logic of my system ActStock. The View was responsible for the user interface of the system and Java provided the choice in choosing JavaFX (a Java framework) to build graphical user interfaces (GUI) for the developed system. Last, but not the least, the Controller which was responsible for the management of the interactions between the Model and the View. This could be implemented by utilizing Java's event handling mechanisms.

Moreover, Java's Object Oriented (OO) nature was very beneficial in the designing of the stock management system –ActStock. It made it possible for representing the real-world entities that are involved in the stock management system –ActStock, which was displayed as classes. This method facilitated the modularity, code reuse as well as maintainability.

Also, Java's flexibility and the reliability were very helpful in developing an efficient and robust stock management system. Its huge library support gave a wide range of tools as well as frameworks which the developer could utilize in order to expedite the development of the system. And this programming language is quite compatible across various platforms which made sure that the system can be deployed on different devices and platforms.

6.1.2 Maven

In order to develop the system effectively, I used Maven as an assistant which helps in organizing and managing various parts of the project. As a tool which automate the tasks , with Maven I managed the libraries, frameworks and other external elements my project needed. Maven downloaded everything the project development needed so there were no concerns of lacking anything the development process needed.

Maven helped to streamline the workflow of the development as well. With the help of this, a clear structure of the project was described to me so that I could organize my code and resources in a consistent way.

6.2 DB SQLite

For the database management system of this project, I chose DB SQLite. It is a file-based database engine which was easy to integrate into my Java developed system. This provided me with a simple and very efficient way for storing and retrieving the stock – related information. By utilizing DB SQLite , I was able to design as well as manage the schema of the database, execute the SQL queries and also ensure that the data integrity for the stock management system –Actstock.

6.3 IntelliJ IDEA

IntelliJ IDEA was used as the integrated development environments (IDE) for the development of the project. IntelliJ IDEA is very user-friendly and has powerful features which enhanced the productivity of the development process. This IDE gave tools for the code editing, debugging process and project management process. With IntelliJ IDE I was able to write a great code as it helped me to quickly address any issues if I faced them while developing the system.

By combining these technologies – Java, JavaFX, Maven, DBSQLite and IntelliJ IDEA, I targeted to create an efficient and dependable stock management system for the university. These selections made sure that the development process was flexible, usable, efficient and had an effective management of the stock-related data.

6.4 Sprints

In this part of the chapter implementation, I will focus on the sprints that are described in the Methodology section of this document. I will also focus on how these sprints is being implemented for the project development process. Each and every sprint is implemented separately and with its own requirement sets which needs to be succeeded. Moreover, due to an agile development method, some of the requirements of the system will get evolved before the project ends.

6.4.1 Sprint 1:

During this sprint, I had the focus on obtaining the user requirements from the analysis. I also had the focus on getting the system designing done. And in this sprint the login and registration procedure was also implemented.

Questionnaires were shared with some staff of the university to understand the needs of them and expectations from this new system of stock management –ActStock. Based on the requirements collected, I designed an initial system design which includes the user interface layout and some UML diagrams which described the overall system architecture.

For login and registration user need to enter relevant username and password for access.

6.4.2 Sprint 2:

In this sprint, most of my focus is on developing the core functionality of the stock management system. This sprint involved implementing some features such as adding stock, updating stock and retrieving stock items, managing the quantities and generating the report. The development of the system was done using Java and JavaFX with regard to the rules of Object oriented programming and MVC.

6.4.3 Sprint 3

This sprint was about the user interface (UI) of the stock management system –ActStock. Feedback from the stock managing staff was collected to understand their necessities and preferences more. Based on the feedback, I made necessary adjustments in order to the UI design so that it is more user-friendly and it enhance overall user experience. Moreover, I conducted thorough testing in order to ensure that the stock management system functions effectively and correctly, it performs well and meets the usability standards.

6.4.4 Sprint 4:

This sprint was the final sprint of this project. My main focus was on addressing any bugs or problems the system encounters while in the testing process. Problems were carefully identified and fixed in order to improve the stability and reliability of the system. I took this sprint to optimize the stock management system's performance and made necessary enhancement.

6.5 Implementation Overview:

This project of stock management system was developed or coded in consideration of the Model View Controller (MVC) pattern. Here is a screenshot from my project below:

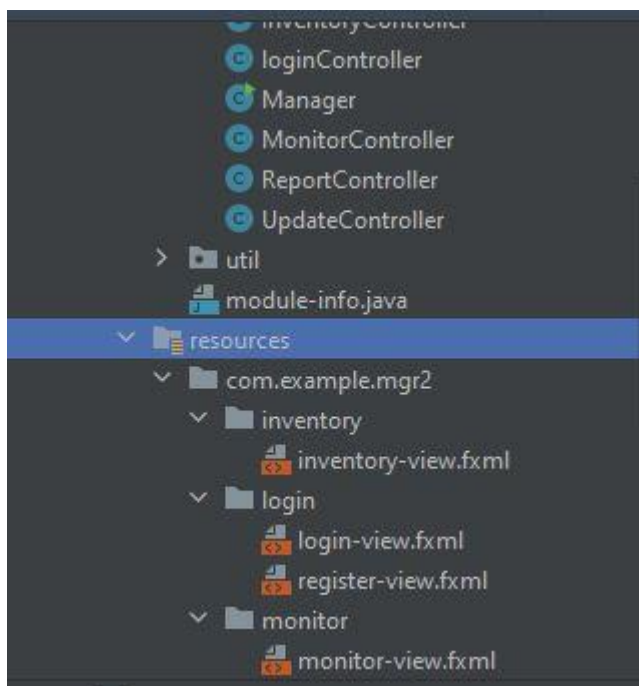


Figure 8MVC pattern in code

6.6 Database of the system:

The database has a table of four. They are:

Item

Usage

User

SQLite_sequence

6.6.1 Item Table:

The SQL statement in this table has several columns, such as the "id", "item_name", "location", "supplier" and "count". The :id: is the Primary Key and item_name is unique. Here is a snippet of the SQL statement:

name	type	constraint
Tables (4)		
Item		CREATE TABLE Item (id TEXT not null constraint Item_pk primary key, item_name TEXT not null constraint Itb
id	TEXT	"id" TEXT NOT NULL
item_name	TEXT	"item_name" TEXT NOT NULL UNIQUE
description	TEXT	"description" TEXT
location	TEXT	"location" TEXT
supplier	TEXT	"supplier" TEXT
count	integer	"count" integer NOT NULL

Figure 9 SQL Item

6.6.2 Usage Table:

This table has columns such as "id:", "user_id", "item_id", "quantity" and "time". This defines the primary key and foreign key constraints for data integrity. SQL statement is below:

Usage		CREATE TABLE Usage (id TEXT not null constraint Usage_pk primary key, user_id TEXT not null constraint Usa
id	TEXT	"id" TEXT NOT NULL
user_id	TEXT	"user_id" TEXT NOT NULL
item_id	TEXT	"item_id" TEXT NOT NULL
quantity	integer	"quantity" integer NOT NULL
time	TEXT	"time" TEXT NOT NULL

Figure 10 SQL usage

6.6.3 User Table

This table has columns such as "id," "user_name," "password," "full_name," and "role." This also specifies a primary key on the "id" and unique constraint on the "user_name" column.

Here is a SQL statement snippet:

✓	User	CREATE TABLE "User" (id TEXT not null constraint User_pk primary key, user_name TEXT not null, password TEXT not null, full_name TEXT, role TEXT)
	id	TEXT "id" TEXT NOT NULL
	user_name	TEXT "user_name" TEXT NOT NULL UNIQUE
	password	TEXT "password" TEXT NOT NULL
	full_name	TEXT "full_name" TEXT
	role	TEXT "role" TEXT

Figure 11SQL User

6.6.4 Sqlite_sequence Table:

This table has columns “name” and “seq” in the SQLite database. This is to keep track of the sequence numbers for auto-incrementing columns of other tables. The SQL statement is below:

✓	sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)
	name	"name"
	seq	"seq"

Figure 12SQL sqlite_sequence

6.6.5 Database connection method:

The project used SQLite database to store the data related to stock. Here is a screenshot of the code.

```
package util;
import java.sql.*;

19 usages
public class Database {
    1 usage
    static String URL = "jdbc:sqlite:db.sqlite";

    4 usages
    static Connection connection;

    14 usages
    public static Connection getConnection() throws SQLException {
        if ( connection != null ) {
            return connection;
        }
        connection = DriverManager.getConnection(URL);
        System.out.println("Database Connection Established successfully");
        return connection;
    }
}
```

Figure 13. database connection

6.6.6 Register method:

This method is used to create an account for the user to access the system. Usernames for this method should be unique. The password should match. System should not allow if any of these requirements are not followed.

```
public void register(ActionEvent event) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Registration");
    if (!Validator.validateIsEmpty(new TextInputControl[]{this.uName, this.pWord, this.rpWord, this.fName})
        || duplicateUser(this.uName.getText())) {
        alert.setContentText("Username is taken!");
    } else if (!this.pWord.getText().equals(this.rpWord.getText())) {
        alert.setContentText("Passwords mismatch!");
    } else {
        String sql = "INSERT INTO User values ( ?, ?, ?, ?, ? )";
        PreparedStatement statement = null;
        ResultSet rs = null;

        try {
            statement = Database.getConnection().prepareStatement(sql);
            UUID uuid = UUID.randomUUID();
            statement.setString( parameterIndex: 1, uuid.toString());
            statement.setString( parameterIndex: 2, this.uName.getText());
            statement.setString( parameterIndex: 3, this.pWord.getText());
            statement.setString( parameterIndex: 4, this.fName.getText());
            statement.setString( parameterIndex: 5, this.role.getText());
            statement.executeUpdate();
        } catch (SQLException exception) {
            System.out.println("Database error");
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
            } catch (SQLException exception) {
                // Handle result set close exception
            }
        }
    }
}
```

Figure 14. registration method

6.6.7 Login method:

This method is used to log in the user of the system if the username and entered password are matched.

Here is a screenshot of the code:

```
public boolean login(String user, String pass) {
    String sql = "SELECT * FROM User WHERE user_name = ?";
    PreparedStatement statement = null;
    ResultSet rs = null;

    try {
        statement = Database.getConnection().prepareStatement(sql);
        statement.setString( parameterIndex: 1, user);
        rs = statement.executeQuery();

        if (rs.next()) {
            return rs.getString( columnLabel: "password").equals(pass);
        }
        return false;
    } catch (SQLException exception) {
        System.out.println("Database error");
        return false;
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException exception) {
            // Handle result set close exception
        }
    }
}
```

Figure 15. login method

6.6.8 Error message while attempting to login

This code is used to display a message with error when the user enters irrelevant information or unmatched password and username. Here is the screenshot below:

```
        if (rs.next()) {
            return rs.getString( columnLabel: "password").equals(pass);
        }
        return false;
    } catch (SQLException exception) {
        System.out.println("Database error");
        return false;
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException exception) {
            // Handle result set close exception
            System.out.println("Error closing result set");
        }
        try {
            if (statement != null) {
                statement.close();
            }
        } catch (SQLException exception) {
            // Handle statement close exception
            System.out.println("Error closing statement");
        }
    }
}
```

Figure 16. Error message for attempting to login with wrong data

6.6.9 User Interface of the Register and Login page:

This is the JavaFX code that is use to create the User Interface of the stock managements system – Actstock. Here is the screenshot below:

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<AnchorPane prefHeight="167.0" prefWidth="394.0" xmlns="http://javafx.com/javafx/17.0.2-ea" xmlns:fx="http://javafx.com/fx"
  <children>
    <Label layoutX="30.0" layoutY="62.0" text="Username" />
    <Label layoutX="30.0" layoutY="97.0" text="Password" />
    <PasswordField fx:id="Password" layoutX="111.0" layoutY="93.0" prefHeight="25.0" prefWidth="261.0" />
    <Button layoutX="253.0" layoutY="124.0" mnemonicParsing="false" onAction="#switchToRegister" text="Register">
      <font>
        <Font name="System Italic" size="12.0" />
      </font></Button>
    <Button layoutX="317.0" layoutY="124.0" mnemonicParsing="false" onAction="#authenticate" text="Sign In">
      <font>
        <Font name="System Bold Italic" size="12.0" />
      </font></Button>
    <TextField fx:id="userName" layoutX="111.0" layoutY="58.0" prefHeight="25.0" prefWidth="261.0" />
    <Label layoutX="35.0" layoutY="14.0" text="Enter Username and Password to Login" textAlignment="CENTER">
      <font>

```

Figure 17 User Interface of the Register and Login page:

6.6.10 Inventory Controlling method:

This conducts the “take out” method which handles the retrieving of stock item functionality in the stock management system –ActStock. This performs validations on the input values , for example the count and the selected item. If they are valid, it inserts the retrieving information into the database. It also updates the item count as well as display the appropriate alert message to the user.

This implements the initialize method from the initializable interface. The “items” map is initialized and the items names gets retrieved from the database and is then added to the choice box for the item selection.

This also includes many private helper methods such as the “getUserId” and “getItemNames” which are responsible for the retrieving of data from the database, performing the required calculations and updating the UI components accordingly.

Here are the code screenshots:

```

String ts = sdf.format(timestamp);

if ( getCurrentCount(this.itemDropdown.getValue()) >= Integer.parseInt(this.count.getText()) ){
    try {

        statement = Database.getConnection().prepareStatement(sql);
        UUID uuid = UUID.randomUUID();
        statement.setString( parameterIndex: 1, uuid.toString());
        statement.setString( parameterIndex: 2, getUserId(this.user.getText()));
        statement.setString( parameterIndex: 3, this.items.get(this.itemDropdown.getValue()));
        statement.setInt( parameterIndex: 4, Integer.parseInt(this.count.getText()));
        statement.setString( parameterIndex: 5, ts);
        statement.executeUpdate();

        updateCount( newCount: getCurrentCount(this.itemDropdown.getValue()) - Integer.parseInt(this.count

        alert.setAlertType(Alert.AlertType.INFORMATION);
        alert.setContentText("Items successfully checked out!");

    } catch (SQLException exception) {
        alert.setContentText("Database error");
    } catch (NumberFormatException exception) {
        alert.setContentText("Count should be a number");
    } finally {

```

Figure 18 Inventory Controlling method

6.6.11 Monitor controller method:

This is responsible for managing as well as displaying alerts that are related to low stock levels in the stock management system – ActStock. This initializes a table view with columns of item name, count and alert level. This retrieves data from the database, create database, create AlertData objects and populated the table with the necessary data. Here is the screenshot of the code:

```

public void initialize(URL url, ResourceBundle resourceBundle) {
    itemCol.setCellValueFactory(new PropertyValueFactory<AlertData, String>(s: "item"));
    countCol.setCellValueFactory(new PropertyValueFactory<AlertData, Integer>(s: "count"));
    alertCol.setCellValueFactory(new PropertyValueFactory<AlertData, String>(s: "alert"));

    table.setItems(getTableData());
}

1 usage
private ObservableList<AlertData> getTableData() { return FXCollections.observableArrayList(getData()); }

1 usage
private List<AlertData> getData() {
    String sql = "SELECT * FROM Item WHERE count < 10 order by count";
    PreparedStatement statement = null;
    ResultSet rs = null;
    List<AlertData> list = new ArrayList<>();

    try {
        statement = Database.getConnection().prepareStatement(sql);
        rs = statement.executeQuery();

        while (rs.next()) {
            AlertData alertData = new AlertData();
            alertData.setItem(rs.getString(columnLabel: "item_name"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return list;
}

```

Figure 19 Monitor Controlling Method

6.6.12 Report Controller Method:

This is responsible for report generating based on the data of the item from database and displaying it in a table. This also provides a functionality to download the report as a CSV file. This initializes a table view with columns for the item name, stock level, usage and the popularity of the item. This retrieves data from the database, calculate the popularity using the stock level and usage. Then the table is populated. Here is the snippet of the code:

```

while (rs.next()) {
    ReportData reportData = new ReportData();
    reportData.setItem(rs.getString( columnLabel: "item_name"));
    reportData.setStockLevel(rs.getInt( columnLabel: "count"));
    reportData.setUsage(rs.getInt( columnLabel: "all_taken"));
    if (reportData.getStockLevel() == 0) {
        reportData.setPopularity("100%");
    } else {
        reportData.setPopularity((reportData.getUsage() / reportData.getStockLevel()) + "%");
    }
    list.add(reportData);
}
return list;
catch (SQLException exception) {
    return null;
finally {
    try {
        if (rs != null) {
            rs.close();
        }
    } catch (SQLException exception) {
        // Handle result set close exception
        System.out.println("Error closing result set");
    }
}
}

```

Figure 20 Report Controller method

6.6.13 Update Controller method

This allows the user to add new items or update the count of the existing items. This also retrieves data from the user interface of the system which validates the input and performs the necessary database operations. Here is a snippet of the code:


```

if (this.itemDropdown.getValue() != null) {
    if (!Validator.validateIsEmpty(new TextInputControl[]{this.count})) {
        String sql = "UPDATE Item SET count = ? WHERE item_name = ?";
        PreparedStatement statement = null;

        try {
            statement = Database.getConnection().prepareStatement(sql);
            statement.setInt( parameterIndex: 1,  x: Integer.parseInt(this.count.getText()) + getCurrentCount(
            statement.setString( parameterIndex: 2, this.itemDropdown.getValue());
            statement.executeUpdate();

            alert.setAlertType(Alert.AlertType.INFORMATION);
            alert.setContentText("Items Added!");

        } catch (SQLException exception) {
            alert.setContentText("Database error");

        } catch (NumberFormatException exception) {
            alert.setAlertType(Alert.AlertType.ERROR);
            alert.setContentText("Count should be a number");

        } finally {
            try {
                if (statement != null) {
                    statement.close();

```

Figure 21 Update Controller method

6.6.14 Controller of the View switching and User Management:

This contains methods which switch between different views such as main view, inventory view, update view , monitor view and report view. This also provides functionality in order to set the user who is using the system currently and handles the switching to the login view.

Here is a screenshot of the code below:

```

public Controller() { /* compiled code */ }

public void setUser(java.lang.String user) { /* compiled code */ }

public void switchToMain(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

1 usage
public void switchToInventory(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

1 usage
public void switchToUpdate(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

1 usage
public void switchToMonitor(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

1 usage
public void switchToReports(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

public void switchToLogin(javafx.event.ActionEvent event) throws java.io.IOException { /* compiled code */ }

```

Figure 22Controller of the View switching and User Management

6.6.15 JavaFX Application Manager

This extends the 'Application' class and overrides the start method in order to setup the initial stage and scene. The application then launches with the login view and the stage is displayed with a fixed size of pixels. The stage is titled as "Inventory Manager". Here is the snippet of the code:

```
package com.example.mgr2;

public class Manager extends javafx.application.Application {
    public Manager() { /* compiled code */ }

    public void start(javafx.stage.Stage stage) throws java.io.IOException { /* compiled code */ }

    public static void main(java.lang.String[] args) { /* compiled code */ }
}
```

Figure 23Manager

6.7 Testing:

6.7.1 Sprint 1:

Login and Register

Test	Expected Outcome	Pass/Fail
Display login page	Login page displayed	Pass
Prompt the user to register	User registration prompt is shown	Pass
Validate user login credentials	User is granted access if credentials match	Pass
Show error message if credentials not met	Error message pops up and user access is rejected.	Pass

Register or create an account	User with all the relevant information and password is able to create an account successfully	Pass
-------------------------------	---	------

Table 12 Login and Register test

Database

Test	Expected Outcome	Pass/Fail
Database should be created and connected	SQL file should be created in the same directory as the program.	Pass
Insert data from system into relevant columns of the database	Data should be stored in the database in the relevant columns successfully	Pass

Table 13 Database test

6.7.2 Sprint 2:

Home Page

Test	Expected Outcome	Pass/Fail
Show the Homepage with the main Functions	Display the main functions	Pass
Display the signed-in username	User's username shown on the Homepage	Pass

Table 14 HomePage test

Update Controller

Test	Expected Outcome	Pass/Fail
Add new items to the stock	New items are successfully added to stock	Pass
Update existing items in stock	Existing items are updates in the stock	Pass

Table 15 Update controller test

Inventory Controller

Test	Expected Outcome	Pass/Fail
------	------------------	-----------

Take out items from stock	Items are successfully retrieved from the stock.	Pass
Update item count after retrieving	Item count is updated correctly	Pass

Table 16Inventory Controller test

6.7.3 Sprint 3:

Monitor Controller

Test	Expected Outcome	Pass/Fail
Show Alerts for the low stock items	Items are displayed for the low stock items.	Pass
Assign appropriate alert levels	Alert levels are assigned correctly	Pass

Table 17Monitor Controller test

Report Controller

Test	Expected Outcome	Pass/Fail
Generate report with the item details	Report includes item name, usage, stock popularity	Fail
Download the report in CSV format	Downloaded successfully as CSV	Fail

Table 18Report Controller test 1

6.7.4 Sprint 4:

Report Controller (as it failed in sprint 3)

Test	Expected Outcome	Pass/Fail
Generate report with the item details	Report includes item name, usage, stock popularity	Pass
Download the report in CSV format	Downloaded successfully as CSV	Pass
Check for bugs in the whole system	No bugs found. Able run all functions as expected	Pass

Table 19Report Controller test2

7. Project evaluation

This chapter is going to evaluate the choices that were made and utilized in this project of stock management system. Moreover, this chapter will also reflect on how the project development was successful and its performance. Lastly, this evaluation chapter will also focus on the expandability and the improvements which could further develop the stock management system.

7.1 Choice of technologies

The choice of using Java as the programming language was very beneficial for the project. Java's versatility and the great adoption made it an ideal choice for the stock management system development. It gave a robust platform in order to implement the system's functionalities and ensured the compatibility across various operating systems.

SQLite database was a reliable and scalable solution for the storing purpose and the retrieving purpose of the stock-related data. The SQLite integration with Java made sure that seamless data management occurred within the system.

The use of Maven as a building tool made the development process of the project easy as it simplifies the project's dependency management and allowed for easy integration of libraries which are external.

IntelliJ IDE was selected as the development environment of the stock management system-ActStock. It gave a user friendly interface to the development of the project, code analysis tools and debugging capabilities as well as enhances the development workflow and productivity.

7.2 Development of the system:

Using the agile methodology for the development of the stock management system was successful. It helped to keep my project on track, with all those procrastinating of mine, and also it allowed to add new necessities to the system built as it was continuing to develop. Using of sprints made it quite easy in order to develop the system's JavaFX user interface and backend functionality separately and then combine them seamlessly.

More time could have been used on the research about the design of this project. And also the database could have been made sure that it has no data duplication entries by normalization process.

Some of the low priority requirements like the requirement to integrate the university's finance system in the stock management system were not implemented this time. If added, this requirement would have added more functionality to the system and would have made it easier for the user to use this stock management system – ActStock. And all the functional and non-functional requirements which were in the 'MUST' for the project were accomplished.

7.3 Performance and Limitations of the system

This Java built project performed very well with all the necessary requirements being applied. But, in the development, not being able to get much research papers which is about stock management systems in the universities made it quite difficult to decide on what all requirements would be possible to implement and what all functions are required in the system if it needs to be an efficient one for a university use. Also, I could have been able to know how to implement those functions easily into the system.

7.4 Improvements and Expandability of the system

Some of the expandability for this project that would enhance the functionality would be to have the ability where the other staff of the university (academic staff/admin staff) could be able to submit requests for taking an item from the stock.

One of the improvement would be to have the university finance system integrated into the stock management system. The finance system when integrated can allow the stock management system to have a function on cost analysis, budget allocation and finance forecasting according to the stock data in the university.

7.5 The learning Outcomes

Throughout the course of this project, my main focus was on enhancing my skills in coding with Java and in an object-oriented method. This project has given me with a comprehensive opportunity to delve into different aspects of Java desktop app creation and has critically contributed to expanding my knowledge in this domain. Due to this, I am eager to continue my journey of learning and growth in this field.

One of the most important learnings from this project has been the invaluable understanding of user interface (UI) design. Developing ActStock, the stock management system for university particularly in this case Islamic University of Maldives, has allowed me to grasp the significance of creating user-friendly interfaces which are visually appealing and easy to navigate.

7.6 My Self Reflection

This project I can say is one of the most time taken project in my three years of Degree in Computer Science. Before this project, I had always done small coding projects in Java using the Netbeans IDE. Or I had done group projects if it has many functions involved in the system. In the development of this stock management system, I learnt how to manage time while working and studying for other modules. Also, I managed the first two sprints well and to be honest I fell behind at the end sprints. The knowledge I got from this project will always be remembered and I never knew that doing researches before developing a project is this necessary in my whole life.

8. Conclusion

The project was a success in managing the stocks of a university. The system managed to add, retrieve, show alerts and popularity of the stock items according to the level of items in the stock. Almost all of the functional requirements were implemented in this project and were met to the user expectation.

The ideas I had before starting this project, was successfully accomplished by the end of the sprints. The ActStock I provided with this document is a prototype model with the functionality and not at all focused on the labour cost.

References/Bibliography

1. Ali Khan, J., Ur Rehman, I., Hayat Khan, Y., Javed Khan, I., & Rashid, S. (2015). *Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique. International Journal of Modern Education and Computer Science*, 7(11), 53-59.
2. Tight, M. (2011). *How many universities are there in the United Kingdom? How many should there be? Higher Education*, 62, 649-663. <https://doi.org/10.1007/s10734-011-9411-5>
3. Feder, T. (2018). *Contract lecturers are a growing yet precarious population in higher education. Physics Today*, 71(11), 22-23. <https://doi.org/10.1063/PT.3.4065>
4. Jiang, C. (2010). *Integrating the Use of Spreadsheet Software and VBA in Inventory Simulation. Journal of Software*, 5, 498-505.
5. Sharp, D. (1998). *Reducing avionics software cost through component-based product line development. In Proceedings of the 17th DASC. AIAA/IEEE/SAE Digital Avionics Systems Conference. Bellevue, WA: IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/739846*
6. Lumba, E., & Waworuntu, A. (2022). *Implementation of Model View Controller Architecture in Object Oriented Programming Learning. International Journal of New Media Technology*.

7. Damilola, O. (2023). MVC Architecture. Retrieved from <https://www.educative.io/answers/mvc-explained>
8. von der Maßen, T., & Lichter, H. (2002). Modeling Variability by UML Use Case Diagrams. In *Proceedings of the International Workshop on Requirements Engineering for Product Lines*. Essen, Germany: IEEE. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.2206&rep=rep1&type=pdf#page=25>
9. Galitz, W. (2007). *The essential guide to user interface design*. Indianapolis, IN: Wiley Pub.
10. Sherrell, L. (2013). Waterfall Model. In A.L.C. Runehov & L. Oviedo (Eds.), *Encyclopedia of Sciences and Religions*. Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-8265-8_200285
11. Doshi, D., Jain, L., & Gala, K. (2021). Review of the Spiral Model and Its Applications. *International Journal of Engineering, Applied and Science Technology*, 5(12). DOI: 10.33564/IJEAST.2021.v05i12.053