**BOOKCART**

**SUBMITTED BY**

**Vighnesh Raje and Nikhil Rathod**

**[Seat No. ]**

DATE OF SUBMISSION

28th February-2017

Under the guidance of **Prof. RAKHEE RANE**

Submitted in partial fulfilment of the requirements for qualifying

B.Sc.I.T. (Semester VI) Examination

**THE KELKAR EDUCATION TRUST’s**

**V.G. VAZE COLLEGE OF ARTS, COMMERCE AND SCIENCE**

**Mithagar Road, Mulund (East), Mumbai – 400 081**

**The Kelkar Education Trust’s**

**V.G. VAZE COLLEGE OF ARTS COMMERCE AND SCIENCE**

**Mithghar Road Mulund (E), Mumbai -400 081.**

**CERTIFICATE**

This is to certify that the Project done at **IT LAB** by Vighnesh Raje and Nikhil Rathod

(Seat No. ) and in partial fulfilment for B.Sc.It Degree Examination

has been found satisfactory. This report had not been submitted for any other examination

and does not form part of any other course undergone by the candidate.

**Project Guide H.O.D. External Examiner**

**ACKNOWLEDGEMENT**

**The presented project, as a part of the curriculum was a first of its kind experience for us.**

**We had looked upon this project not merely as an syllabus to be completed but as an aim to**

**know, study, develop and experience the commercial software technologies.**

**we are glad to say that, we have satisfactorily reached our aims and intentions, to make this**

**project a success.**

**we would like to thank and appreciate the support of few, who served a helping hand**

**physically, mentally and intellectually in the course of this project.**

**Foremost regards to my guide Prof.Rakhee Rane**

**we would thank our H.O.D, Prof K.G. Bhole, who made available the facilities required for the**

**project work.**

**we also wish to mention the unsaid support of my parents who, as always helped me in every**

**possible way to make this work of mine, a success.**

**The contribution made by my friends and mates, directly or indirectly was indispensable, and**

**will always be remembered.**

**This opportunity has given us a valuable experience about software development.**

**BOOKCART**

**INDEX**

|  |  |
| --- | --- |
| **Topic** | **Page No.** |
| **CHAPTER 1 : SYNOPSIS**   * 1. **TITLE**   2. **STATEMENT ABOUT PROBLEM**   3. **WHY THIS TOPIC**   4. **OBJECTIVE AND SCOPE**   5. **PROPOSED ARCHITECTURE**   Fig 1.1 Proposed Architecture   * 1. **PROPOSED ARCHITECTURE**   2. **REQUIREMENT**   3. **TESTING TECHNOLOGY**   **1.9 CONTRIBUTION** | **6**  **9**  **9** |
| **CHAPTER 2 : THEORETICAL BACKGROUND**  **2.1 INTRODUCTION OF PROJECT**  **2.2 EXISTING SYSTEM**  **2.2.1 Disadvantages Of Existing System**  **2.3 PROPOSED SYSTEM**  **2.3.1 Advantages Of Proposed System** |  |
| **CHAPTER 3: FUCTIOANAL, NON FUNCTIONAL AND SYSTEM REQUIREMENTS**  **3.1 USER REQUIREMENTS**  **3.2 FEASIBILITY STUDY**  3.2.1 ECONOMIC FEASIBILITY  3.2.2 OPERATIONAL FEASIBILITY:  3.2.3 TECHICAL FEASIBILITY  3.2.4 SCHEDULED AND RESOURCEFEASIBILITY  **3.3 DIAGRAMS**  Table 3.1 Notations of Use Case Diagram   1. **Use Case Diagram:**   Fig. 3.1 Use case diagram  **B Activity Diagram:**  Table 3.2 ACTIVITY DIAGRAM Notations  Fig. 3.2: Activity Diagram for user process   1. **Sequence Diagram:**   Table 3.3 Notation for sequence diagram  Fig. 3.3 Sequence Diagram for registration process  Fig. 3.4 Sequence Diagram for login process  Fig. 3.5 Sequence Diagram for payment process  Fig. 3.6 Sequence Diagram for Login failed process  Fig. 3.7 Sequence Diagram for Orders process  Fig. 3.8 Sequence Diagram for Return process   1. **Class Diagram :**   Table 3.4 Notation for class diagram  Fig. 3.9 Class Diagram for the System |  |
| **CHAPTER 4: TESTING**  **4.1 TESTING**  **4.2 TYPES OF TESTING**  4.2.1 WHITE-BOX TESTING  4.2.2 BLACK-BOX TESTING  4.2.3 FUNCTIONAL TESTING  4.2.4 PERFORMANCE TESTING  4.2.5 INTEGRATION TESTING  4.2.6 UNIT TESTING  4.2.7 SYSTEM TESTING  **4.3 TESTING METHADOLOGY**  **4.4 TEST CASES** |  |
| **CHAPTER 5 : TOOLS FOR DEVELOPING SYSTEM** |  |
| **CHAPTER 6: COST AND BENEFIT ANALYSIS**  **6.1 COST ANALYSIS**  **6.2 PROJECT COST USING ‘COCOMO’** |  |
| **CHAPTER 7: LIMITATIONS**  **CONCLUSION** |  |

**Chapter 1. Synopsis**

**1.1 TITLE:** Bookcart

**1.2 STATEMENTS ABOUT THE PROBLEM:** Books are integral part of everyone’s life. Even if you consider period of year we put our hands on so many books but how many of these books we consider reading twice? In this thought people consider issuing book from library but there too they have to wait till a copy is available also they have to check different libraries to check for availability.

**1.3 WHY THIS TOPIC IS CHOSEN:**

We suddenly came up with this idea. Most of us face the above problem so we thought something like this would be helpful to some extent.

**1.4 OBJECTIVES:**

* User has to register by providing required details
* Choose a yearly subscription plan
* Search for the books
* Request for delivery of the address provided during registration
* After reading book request for return

**1.5 SCOPE:**

Only available for users in Thane and Mulund.

**1.6 PROPOSED ARCHITECTURE:**

CLIENT-SERVER ARCHITECTURE

It is network architecture in which each computer or process on the network is either client or a server .Servers are powerful computers or processes dedicated to mapping disk, printers or network, traffic. Clients are PCs or workstations on which users run applications. Clients rely on servers for resources such as files, devices and even processing power

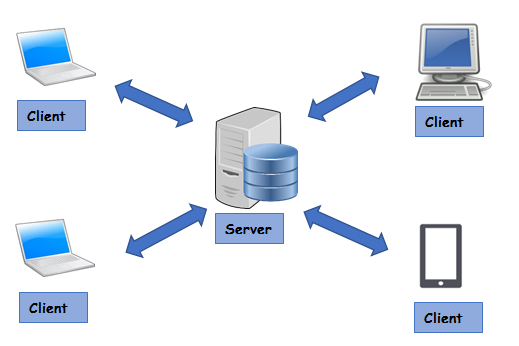


Fig.1.1 Proposed Architecture

**1.7 PROPOSED METHODOLOGY:**

INCREMENTAL BUILD MODEL-

The incremental build model is a method of software development where the product is designed ,implemented and tested incrementally until the product is finished involves both development and maintenance. Product is developed into number of components, each of which is designed and built separately. This model of development helps ease the traumatic effect of introducing a completely new system all of once.

**1.8 REQUIREMENT:**

* HARDWARE-

1. 1.3 GHz processor.
2. 1 GB of ram.
3. Minimum 100 mb of internal storage.

* SOFTWARE-

Android version 4.0(jellybean) or higher version

* TOOLS-

1)Android Studio-

It is official integrated development environment(IDE) to android platform. It is available for download on windows, linux and mac.

2)SQLite database-

It is relational database management system contained in a C programming library. It is not a client server database engine rather it is embedded into end program.

**1.9 CONTRIBUTION:**

The application would contribute to the environment the environment as a single copy of book would be used by many people. This will reduce the requirement for number of copies, thus saving trees. Also reader’s choice of book would be delivered at their doorstep their efforts for looking around for the same would be reduced. Books are to be returned therefore cost of maintaining bookshelf would be saved. Also readers wouldn’t have to stress their eyes by reading e-books.

**CHAPTER 2: FUNCTIONAL, NON FUNCTIONAL AND SYSTEM REQUIREMENTS**

**2.1 INTRODUCTION:**

In System Analysis and Design phase, the whole software development process, the overall software structure and its outlay are defined. Analysis and Design are very important in the whole development cycle process. Any fault in the design phase could be very expensive to solve in the software development process. In this phase, the logical product of the system is developed. The software development process, the software's overall structure and its nuances are defined.

This step refers to the gathering of system requirements, with the goal of determining how these requirements will be accommodated in the system. Extensive communication between the customer and the developer is essential. The developer has to develop the system to meet the client‘s needs. But the developer usually does not fully understand the client‘s problem domain, and the client often does not understand the issues involved in the software systems. This causes communication gap, which can be bridged by gathering information from the client and also making him understand what the system will do.

In this phase, database design and the data structure design etc. are defined for our system.

After designing phase, a software development model is created.

Our first task was to study the existing system, analyze it and find its limitations.

**2.2 RISK ANALSIS:**

Risk Analysis is mostly done at the start of the project. Before collecting all the information we must check whether it is feasible, safe to do the project or to check any risk that may occur during the process. These risks may occur at any point during the process and if proper measures aren’t taken it affects the project in different ways.

There are 3 types of risks:

a. Business

b. Project

c. Product

**2.3 PROPOSED SYSTEM STUDY:**

It is always necessary to study and recognize the problem of the existing system, which will help in finding out the requirements for the new system. A thorough study of the proposed system involves finding different alternatives for a better solution.

This step basically deals with the following different operations:

a. Data Gathering

b. Study of Existing System

c. Analyzing problem

d. Studying various documents

e. Feasibility study for further improvement

**2.3.1 TOOLS AND TECHNIQUES USED FOR INFORMATION GATHERING**

INTERVIEWS:

I prepared some questionnaires so as to study the actual problems being faced by the readers in the current method of functioning. It was a kind of informal interview with the reader.

These were some of the questions I asked:

a. Approximately how much do you spend on reading?

b. How much delivery time do you expect?

c. Average time you take to read books?

d. Do you prefer buying books or issuing it from library?

INTERNET RESEARCH:

The Internet is an indispensable tool for getting most of the doubts cleared, if in case not done so by above methods. Almost any kind of information is available through the Internet at a click of the mouse. The resources which I referred through internet are mentioned in the references section of this Document.

**2.4 USER REQUIREMENT:-**

* Functional requirement-

1. Create account(user account).
2. If user is already register then system should provide login facility by using email and password.
3. For new registration, process should be followed i.e accepting email id, address, password, contact number, name etc.
4. System should provide online payment facility for subscription pack.
5. User must be able to pay for subscription pack by online banking, credit or debit card.
6. Confirm order tab is provided by system where the user can choose delivery time slot.
7. Another tab called bookshelf is provided where user can view ordered books.
8. The return tab is provided to request for return of book.

* Non-functional requirement-

1. Reliability: system should be online at any given time.
2. Usability: system should have a simple and clean user interface. The books should be properly organized according to genre, user should able to order less book in less number of clicks.
3. Performance: system should loaded quickly. Response time should be less.
4. Scalability: The performance of system should not degrade if traffic on system increases in future. System should be scalable to include additional book data in future without massive change.
5. Security: Payment transaction should be properly secured. User data should be encrypted. User database should be secured.

**2.5 FEASIBILITY STUDY:**

Feasibility studies aim to objectively and rationally uncover the strengths and

weaknesses of the existing business or proposed venture, opportunities and threats as

presented by the environment, the resources required to carry through, and ultimately

the prospects for success.

The different types of feasibility are as follows:

a. Economic feasibility.

b. Operational feasibility.

c. Technical feasibility.

d. Scheduled and Resource feasibility.

**2.5.1 ECONOMIC FEASIBILITY:**

For evaluating the effectiveness of a new system, the procedure is to determine the

benefits and savings that are expected from a candidate system and compare them

with costs. In case of a new project, financial viability can be judged on the following

parameters:

a. Total estimated cost of the project.

b. Existing investment by the promoter in any other business.

c. Projected cash flow and profitability.

**2.5.2 OPERATIONAL FEASIBILITY**:

Operational feasibility is a measure of how well a proposed system solves the

problems, and takes advantage of the opportunities identified during scope definition

and how it satisfies the requirements identified in the requirements analysis phase of

system development. The new system has completely user friendly interface. It has

been designed to be pretty intuitive, so that even an inexperienced person can easily

handle the system.

**2.5.3 TECHICAL FEASIBILITY:**

Technological feasibility is carried out to determine whether the company has the

capability, in terms of software, hardware, personnel to handle the completion of the

project. The required hardware and software is present with the client. But there is a

need for the administrator to give a basic understanding regarding the working of the

system and frequently used IT terminology in the software. This process is very easy and requires very little time and effort.

**2.5.4 SCHEDULED AND RESOURCE FEASIBILITY:**

The schedule of the project is actually the periodic progress of the project. This is

carefully monitored. In the resource feasibility, the primary resources are the team

members and their designation, i.e., analysts, technician and the user. Their skills have

to be continuously updated and revised.

1. **SCHEDULE FEASIBILITY:**

Typically this means estimating how long the system will take to develop, and if it

can be completed in a given time period using some methods like payback period.

Schedule feasibility is a measure of how reasonable the project timetable is.

The project is feasible as 3-4 months are required to develop which is appropriate for

this project.

1. **RESOURCE FEASIBILITY:**

This study determines whether the company has sufficient amount of resources

available that the project development requires. The resources may involve skilled

lab or, physical worksite, computing facilities, etc. This involves questions such as

how much time is available to build the new system, when it can be built, whether it

interferes with normal business operations, type and amount of resources required,

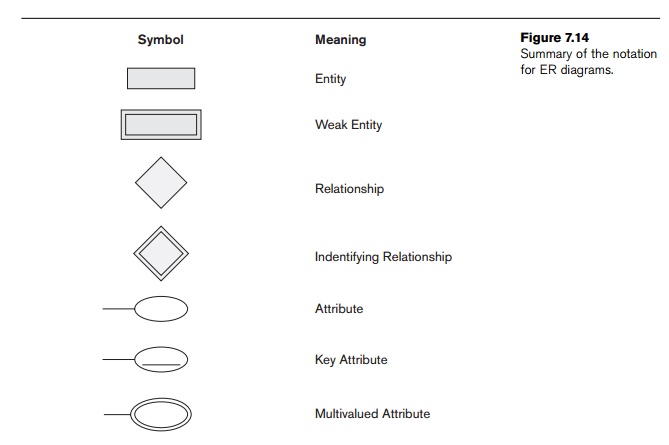
dependencies, etc. The project is feasible because all the necessary resources are

available, but problem may occur.

**2.6 DIAGRAMS:**

**A] E-R diagram:**

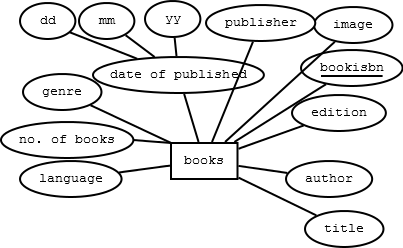
**Notations-**

****

**E-R diagram notation**

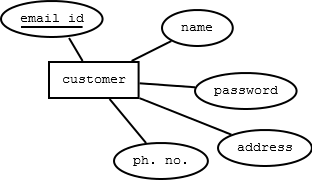
**Entities used in our system:**

**Book entity-**

****

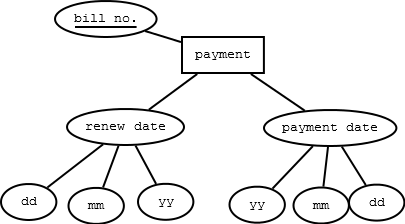
**Book entity**

**Customer entity-**

****

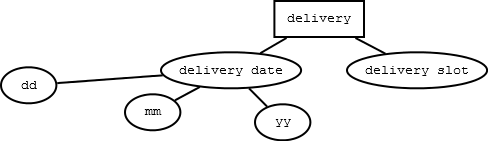
**Customer entity**

**Payment entity-**

****

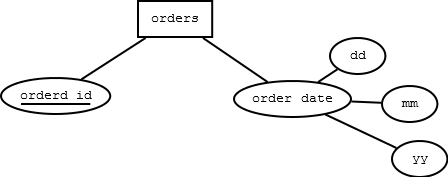
**Payment entity**

**Delivery entity-**

****

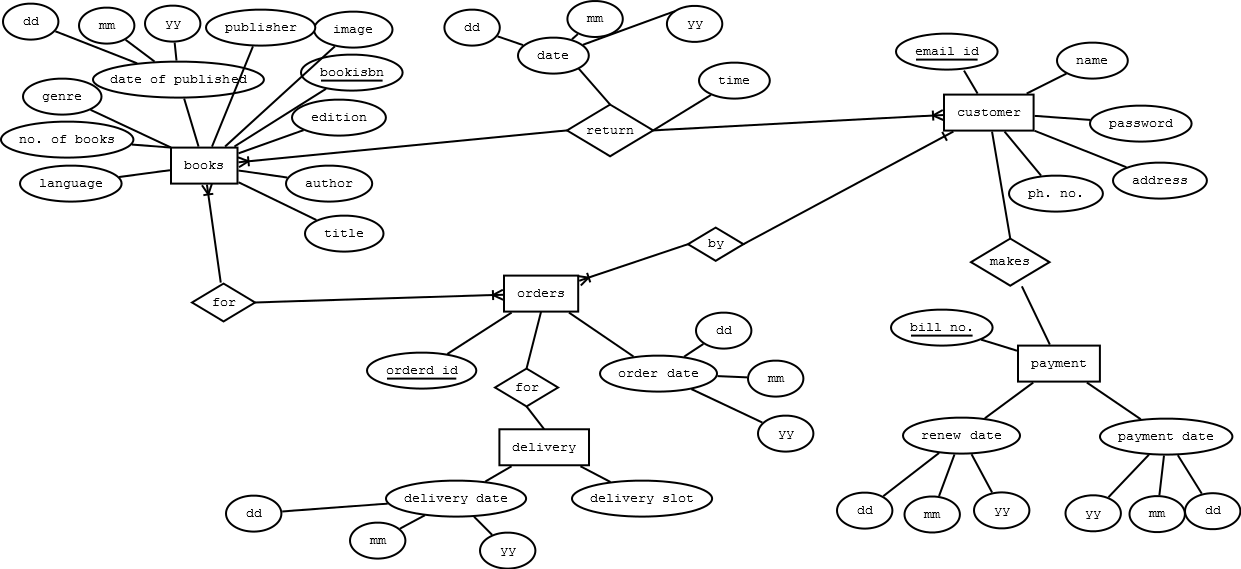
**Delivery entity**

**Order entity-**

****

**Order entity**

**E-R Diagram –**

****

**E-R Diagram**

**B] Use case diagram:**

A use case is a list of steps, typically defining interactions between a role (known in UML as an “actor”) and a system, to achieve a goal. The actor can be a human or an external system.

| **Icon** | **Name** |
| --- | --- |
|  | [Use Case](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-use-case) |
|  | [Association](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-association) |
|  | [Actor](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-actor) |
|  | [System](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-system) |
|  | [Include](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-include) |
|  | [Extend](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-extend) |
|  | [Dependency](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-dependency) |
|  | [Generalization](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-generalization) |
|  | [Realization](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-realization) |
|  | [Collaboration](https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/84257_usecasediagr.html#uml-collaboration) |

Table 3.1 Use case Diagram Notations

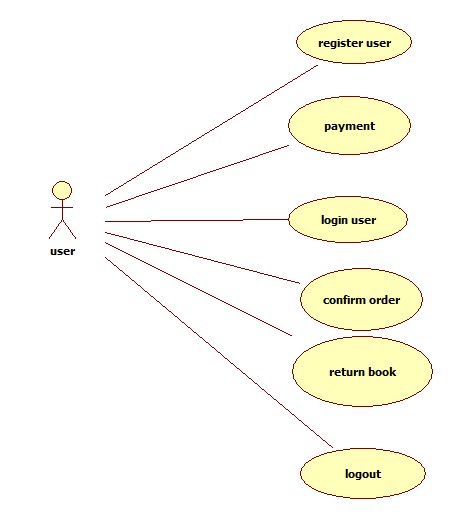


Fig 3.1 Use Case Diagram

1. Use case : Login

Summary: The user gives the details in order to login.

Actor: User

Precondition: The system is waiting of user to login by emailed and password

Description: Login page appears and user has to fill required details to login to system.

Exception:

1. If user does not fills the mandatory details, the login does not take place and system asks the user to refill the details.
2. If the Internet speed is too slow, then successful login of particular user will not take place.

Post-condition: user is logged in to the system and is ready to explore the system.

1. Use case: Register

Summery: the user gives detail information in order to get register.

Actor: User

Precondition: The user must provide all the required information to register themselves to system.

Exception:

1. If user does not provide mandatory information then registration does not take place.
2. If Internet is too slow then registration will not take place .

Post-condition: User is register to system and ready to login.

1. Use case: confirm order

Summery: The user select book and confirm it for delivery.

Actor: User.

Precondition: The user must do payment during registration in order to confirm order for deliver.

Exception:

1. If the payment is not done then user can not confirm the order.

Post-condition: the users order is take place and ready to deliver.

1. Use case: Return book.

Summery: User can request for return of book after reading it.

Actor: User.

Pre-condition : Firstly the book must be deliver to user in order to return.

Exception:

1. If the book is damaged then user has to pay for it.

Post-condition: Once return request is placed the book will get picked up from users register address.

1. Use case: Logout

Summery: User will get back to login screen.

Actor: User.

Precondition: User must logged in in order to logout of the system.

Exception:

1. If the Internet is slow then user log out can not be done.

Post-condition: User is logged out of the system.

1. Use case: payment

Summery: User have to pay to use the system

Actor: user

Precondition: user must pay the required amount to use the system

Exception:

1. User have to pay only by Debit card

Post-condition: once payment is done user can use the system

**C**] **Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and  
actions with support for choice, iteration and concurrency.

**Notation:**

|  |  |  |
| --- | --- | --- |
| Free-form transition link |  | A *transition link* represents control flow between nodes |
| Rounded Rectangle |  | It represents the activity |
| Diamond |  | A logic where a decision is to be made is depicted by a diamond |
| Initial Activity |  | This shows the starting point or first activity of the flow. |
|  |  |  |
| Final Activity |  | The end of the Activity diagram, also called as a final activity. |

Table 3.2 Activity Diagram Notations

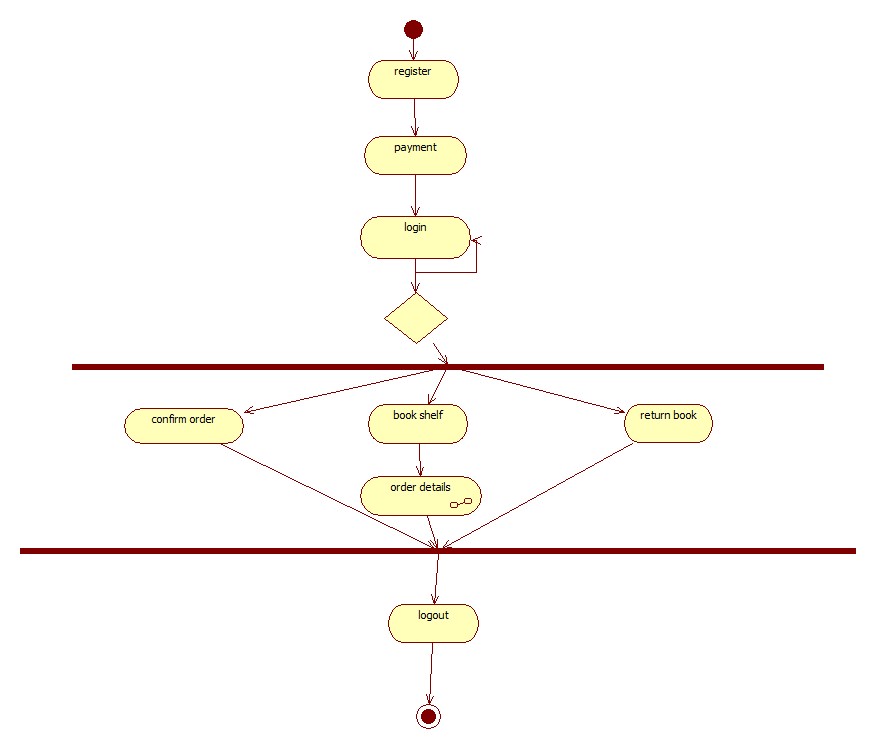
****

Fig 3.2 Activity Diagram

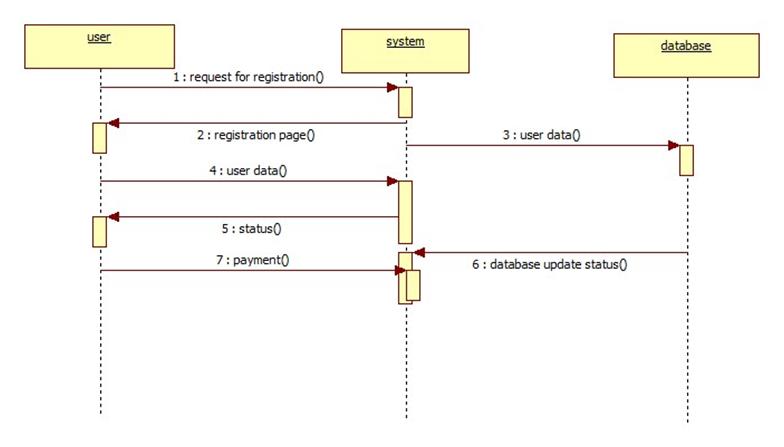
**D] Sequence Diagram:**

A **sequence diagram** in a Unified Modeling Language (UML) is a kind of  
interaction diagram that shows how processes operate with one another and in what order. A  
sequence diagram shows object interactions arranged in time sequence. It depicts the objects  
and classes involved in the scenario and the sequence of messages exchanged between the  
objects needed to carry out the functionality of the scenario. Sequence diagrams typically are  
associated with use case realizations in the Logical View of the system under development.

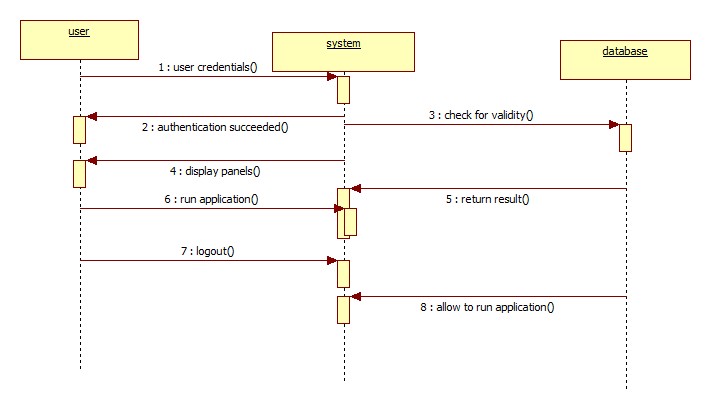
**Notation:**

|  |  |  |  |
| --- | --- | --- | --- |
| Synchronous message | An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result. |  |  |
| Activation | The period during which an object is performing an action |  |  |
| Object instance | An object that is created, performs actions, and/or is destroyed during the lifeline. |  |  |

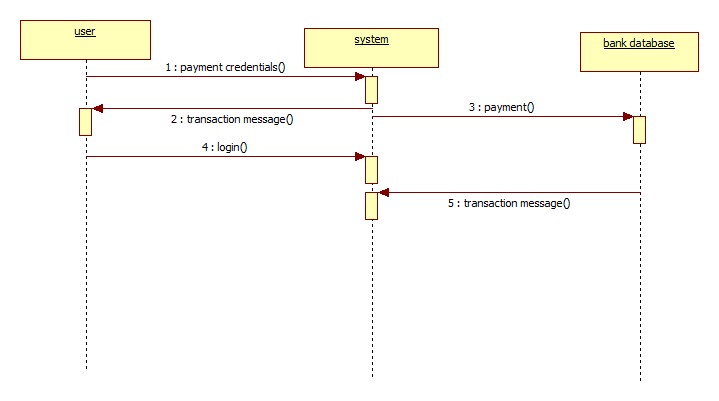
Table 3.3 Sequence Diagram Notations

****

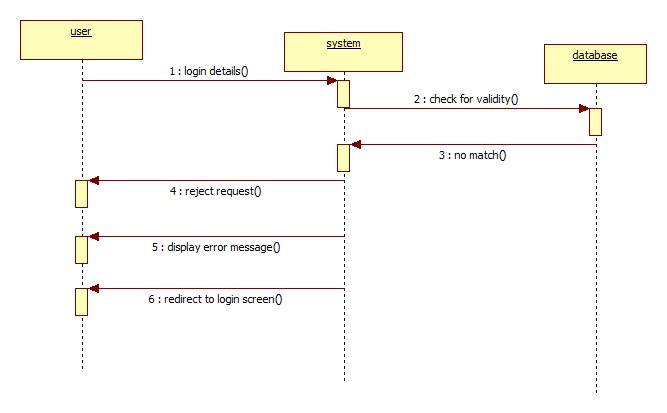
**Fig 3.3 Registration Sequence Diagram**

****

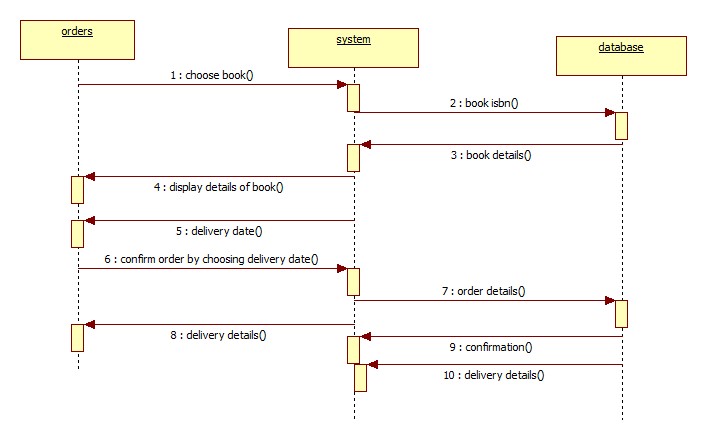
**Fig 3.4 Login Sequence Diagram**

****

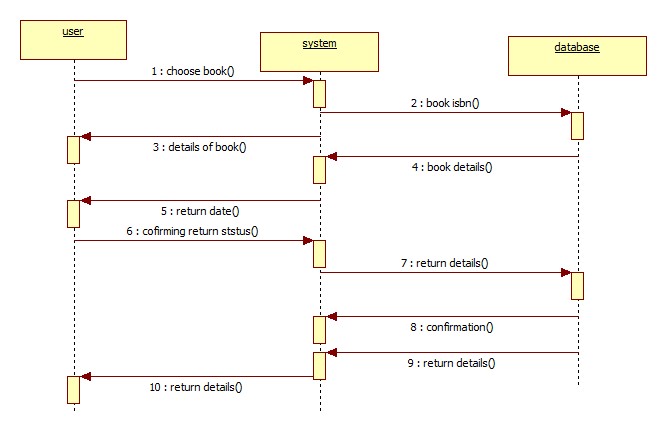
**Fig 3.5 Payment Sequence Diagram**

****

**Fig 3.5 Login Failure Sequence Diagram**

****

**Fig 3.6 Orders Sequence Diagram**

****

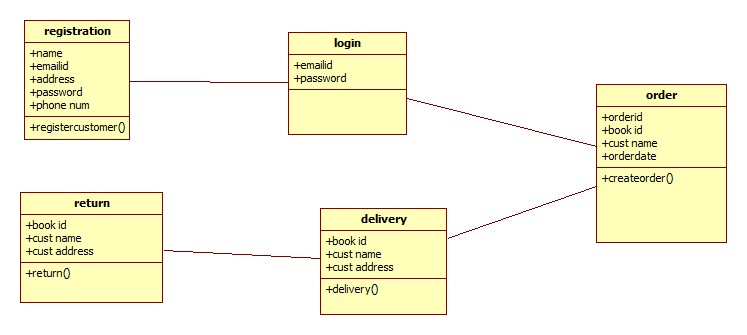
**Return sequence diagram**

**E] class diagram:**

Notation:

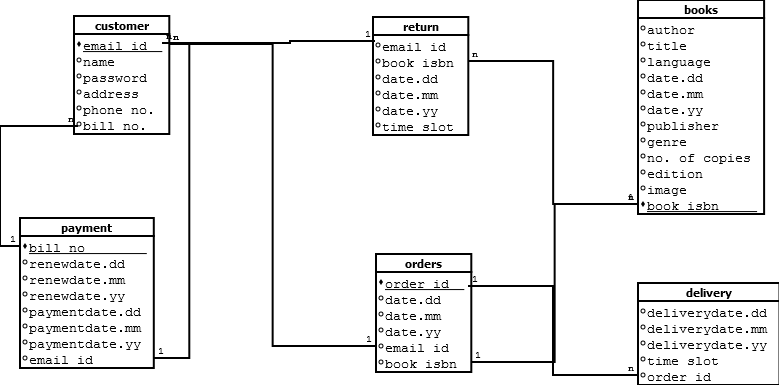
|  |  |  |  |
| --- | --- | --- | --- |
| Sr no. | Name | Description | Symbol |
| 1. | Association | Represents the static relationship shared among the objects of two classes |  |
| 2 | Aggregation | It is an association that represents a part-whole or part-of relationship. |  |
| 3 | Composition |  |  |
| 4 | Generalization | Indicates that one of the two related classes is considered to be a specialized form of the other |  |
| 5 | Dependency | It is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time |  |

Table 3.4 Class Diagram Notations



**Fig 3.9 Class Diagram**

DATABASE SCHEMA –

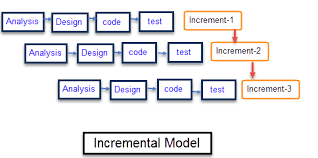


**Chapter 3: SYSTEM IMPLEMENTATION**

**7.1 METHODOLOGY ADOPTED: INCREMENTAL BUILD MODEL -**

Incremental Build Model:

The **incremental build model** is a method of [software development](https://en.wikipedia.org/wiki/Software_development) where the product is [designed](https://en.wikipedia.org/wiki/Software_design), implemented and [tested](https://en.wikipedia.org/wiki/Software_testing)incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the [waterfall model](https://en.wikipedia.org/wiki/Waterfall_model) with the iterative philosophy of [prototyping](https://en.wikipedia.org/wiki/Software_prototyping).



Why this Model is selected?

As all the requirements for my project are well defined & well understood and also I thought that in future new requirements may arise from the user & this as all is supported by the incremental model, I have selected it.

It is divided in following stages:

# Analysis –

Decision over changes to be implemented in the next stage are tak en here. Only small portions are implemented at a time.

# Design –

It includes:

1) What the interface will look like.

2) How the data will be validated.

# Coding –

Compiling the code to make sure that the code is correct.

# Testing –

It involves running the program by using test data if it is requested. If error occurs then the program is debugged & either code or design is altered / changed.

Advantages & Disadvantages of Incremental Model:

**Advantages:-**

Generates working software quickly and early during software life cycle. More flexible-less costly to change scope & requirements. Easy to test & debug during smaller iteration. Easy to handle risk as they are identified during iteration.

**Disadvantages:-**

Needs good planning & design. Needs clear & complete definition of system. New requirements might needs new equipment & also will increase budget.

**Chapter 4: Testing**

**4.1 TESTING:**

Software testing is a process which is used to measure the quality of software

developed. It is also a process of uncovering errors in a program and makes it a

feasible task. It is useful process of executing program with the intent of finding bugs.

In order to prove that a piece of software works, the software must be tested to

determine if the requirements of the application are met. There are several different

types of tests used throughout the development process. The two main types of testing

are white box and black box testing.

White box test cases are used to test specific paths through the code. At decision

points you can test the boundaries of the decision (boundary testing) and the partitions

of the decision (partition testing).

**4.2 TYPES OF TESTING:**

4.2.1 WHITE-BOX TESTING:

In white-box testing an internal perspective of the system, as well as programming

skills, are used to design test cases. The tester chooses inputs to exercise paths

through the code and determine the appropriate outputs.

4.2.2 BLACK-BOX TESTING:

Black box testing have little or no regard to the internal logical structure of the

system, it only examines the fundamental aspect of the system. It makes sure that

input is properly accepted and output is correctly produced.

4.2.3 FUNCTIONAL TESTING:

Functional tests involve exercising the code with nominal input values which gives

the expected results and boundary values are known.

4.2.4 PERFORMANCE TESTING:

Performance tests are designed to verify response time. If the wrong data is entered

then the system does not allow it and calculations are not performed.

4.2.5 INTEGRATION TESTING:

Integration testing is critical to ensure the functional correctness of the integrated

system.

Integration testing can be divided into two categories. Integration testing is often the

most time consuming and expensive part of testing.

4.2.6 UNIT TESTING:

The first test in the development process is the unit test. The source code is normally

divided into modules, which in turn are divided into smaller units called units. These

units have specific behavior. The test done on these units of code is called unit test.

Unit test depends upon the language on which the project is developed. Unit tests

ensure that each unique path of the project performs accurately to the documented

specifications and contains clearly defined inputs and expected results.

4.2.7 SYSTEM TESTING:

Several modules constitute a project. If the project is long-term project, several

developers write the modules. Once all the modules are integrated, several errors may

arise. The testing done at this stage is called system test. System testing ensures that

the entire integrated software system meets requirements. It tests a configuration to

ensure known and predictable results. System testing is based on process descriptions

and flows, emphasizing pre-driven process links and integration points.

**4.3 TESTING METHADOLOGY:**

There are two basic approaches to testing:

black-box and white-box.

In black-box

testing the structure of the program is not considered. Test cases are decided solely on

the basis of the requirements or specifications of the program or module, and the

internals of the module or the program are not considered for selection of test cases.

White-box testing, on the other hand is concerned with testing the implementation of

the program. The intent is not to exercise all the different input or output conditions

(although that may be a by-product) but to exercise the different programming

structures and data structures used in the program.

‘White box testing approach not required to test this website.’

Because, these website restricts user to enter wrong values, if user try to

entered wrong value, website won’t allow to enter that value.

Therefore, white box testing is not necessary.

‘Black box testing approach is used to test this website.’

**BLACK-BOX TESTING:**

In this testing, the structure of the program is not considered. Test cases are decided

solely on the basis of the requirements of specifications of the program or module,

and the intervals of the module or program are not considered for the selection of test

cases.

In black box testing, the tester only knows the inputs that can be given to the system

and what output the system should give. In other words, the basis for declining test

cases in functional testing is the requirements or specifications of the system or

module. This form of testing is also called functional or behavioral testing.

There are various levels of testing which are as follows:

a. Unit testing: Unit testing is essentially for the verification of the codeproduced

during the code phase.

b. System testing: The entire software is tested.

c. Acceptance testing: The external behaviour of the system is focused.

**4.4 TEST CASES**:

If there is a fault in a program, the program can still provide the expected behaviour

for many inputs. Only for the set of inputs that exercise the fault in the program will

the output of the program deviate from the expected behavior. The two fundamental

goals of a practical testing activity are — maximize the number of errors detected and

minimize the number of test cases (i.e., minimize the cost). Hence, an ideal test

caseset is one that succeeds (meaning that its execution reveals no errors) only if there

are no errors in the program.

**SYSTEM MAINTAINANCE:**

Software maintenance is defined in the IEEE Standard for Software Maintenance,

IEEE 1219 [IEEE 1219], as the modification of a software product after delivery to

correct faults, to improve performance or other attributes, or to adapt the product to a

modified environment.

Software maintenance is a very broad activity that includes error corrections,

enhancements of capabilities, deletion of obsolete capabilities, and optimization.

Because change is inevitable, mechanism must be developed for evaluation,

controlling and making modifications. So any work done to change the software after

it is in operation is considered to be maintenance work. The purpose is to preserve the

value of software over the time. The value can be enhanced by expanding the

customer base, meeting additional requirements, becoming easier to use, more

efficient and employing newer technology. Maintenance may span for 20 years,

whereas development may be 1-2 years. Maintenance is not a part of software

development, but is an extremely important activity in the life of a software product.

Maintenance includes all the activities after installation of the software that is

performed to keep the system operational and updated.

Most of the maintenance effort is used for non-corrective actions like functionality

enhancements or inclusion of newer technologies.

The maintenance activities can be categorized into four classes:

1. Adaptive – dealing with changes and adapting in the software environment.
2. Perfective – accommodating with new or changed user requirements which concern functional enhancements to the software.
3. Corrective – dealing with errors found and fixing it.

d. Preventive – concerns activities aiming on increasing software maintainability and

prevent problems in the future.

It may happen that the software may fail after development. A software product wears

out due to age. In software, bugs or errors may get introduced during the design and

development process which may go undetected during the Testing phase. In such

cases, software may fail after operating correctly for some time. Such errors need to

be removed.

So as not to encounter such problems in the future it is necessary to visit the

organization frequently to check whether the system is maintained properly and is

error free.

There is less probability of such problems as the system is developed by considering

the requirements and detailed study of the system. If any problem arises the problem

can be reported by the user and by visiting the place and making changes will keep

the software maintained.

TOOLS USED FOR DEVELOPING SYSTEM

Android studio:



**Android Studio** is the official [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for [Google](https://en.wikipedia.org/wiki/Google)'s [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) [operating system](https://en.wikipedia.org/wiki/Operating_system), built on [JetBrains](https://en.wikipedia.org/wiki/JetBrains)' [IntelliJ IDEA](https://en.wikipedia.org/wiki/IntelliJ_IDEA) software and designed specifically for [Android development](https://en.wikipedia.org/wiki/Android_software_development).It is available for download on [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) based operating systems. It is a replacement for the [Eclipse Android Development Tools](https://en.wikipedia.org/wiki/Eclipse_(software)#Android_Development_Tools) (ADT) as primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the [Google I/O](https://en.wikipedia.org/wiki/Google_I/O) conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.0 released in October 2017.

System requirements:

|  |  |
| --- | --- |
| **Criterion** | **Description** |
| **OS version** | Microsoft® Windows® 7/8/10 (32-bit or 64-bit) Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra) GNOME or KDE desktop Linux (64 bit capable of running 32-bit applications)(GNU C Library (glibc) 2.19+) |
| **RAM** | 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator |
| **Disk space** | 2 GB of available disk space minimum,  4 GB recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) |
| **Java version** | Java Development Kit (JDK) 8 |
| **Screen resolution** | 1280×800 minimum screen resolution |

COST ANALYSIS

**6.1 COST ANALYSIS:**

The cost of a project is a function of many parameters. Foremost among them

is the size of the project. Other factors that affect the cost are programmer

ability, experience of the developers in the area, complexity of the project, and

reliability requirements. It is also due to the requirements of software,

hardware and human resources.

LOC is beneficial in many ways as follows:

a. It is an easy method to measure effort.

b. The alternative methods to the counting of LOC are also fighting with

problems and weaknesses.

c. In spite of its unreliability for individual programs, it gives reliable

average results, which is crucial especially for huge projects.

**6.2 PROJECT COST USING ‘COCOMO’:**

The Constructive Cost Model (COCOMO) is an algorithmic software cost

estimation model developed by Barry W. Boehm. The model uses a basic

regression formula with parameters that are derived from historical project

data and current project characteristics.

COCOMO was first published in Boehm's 1981 book Software Engineering

Economics as a model for estimating effort, cost, and schedule for software

projects. It drew on a study of 63 projects at TRW Aerospace where Boehm

was Director of Software Research and Technology.

The study examined projects ranging in size from 2,000 to 100,000 lines of

code, and programming languages ranging from assembly to PL/I. These

projects were based on the waterfall model of software development which

was the prevalent software development process in 1981.

COCOMO consists of a hierarchy of three increasingly detailed and accurate

forms. The first level, Basic COCOMO is good for quick, early, rough order

of magnitude estimates of software costs, but its accuracy is limited due to its

lack of factors to account for difference in project attributes (Cost Drivers).

Intermediate COCOMO takes these Cost Drivers into account and

Detailed COCOMO additionally accounts for the influence of individual

project phases.

**ADVANTAGES OF INTERMEDIATE COCOMO:**

The Intermediate COCOMO model can be applied to almost entire software

project for easy and rough implementation during early stages. It can also be

applied at the software component level for obtaining more accurate cost

estimation.

**DISADVANTAGES OF INTERMEDIATE COCOMO**:

A Product with many components is difficult to estimate with Intermediate

COCOMO and secondly the effort multipliers are not dependent on phase.

**BENEFIT ANALYSIS:**

Benefit analysis implies the estimation of the benefit achieved through the

project in monitory terms. Profit earned by selling the product is the main

source of earning monetary benefit. The selling cost is decided by dividing the

development cost, by the value of estimated customer count, which is based on

various factors like type of project, its Usefulness-affordability ratio etc.

As the project is developed considering particular target, profit earning is less.

The Development cost can be earned by selling the project to minimum of 5-6

small business. This is not easy to achieve as the project is not generalized.

**LIMITATIONS**

**Limitation Of This Project:**

a. To work on this application, computer should be connected to Internet.

b. If customer is ordering for the first time he/she must register to application and if he/she is not the new customer they only have to log in.

**CONCLUSION**

**CONCLUSION**

This Application provides facility to order books at home and return them after reading

It reduces readers effort to go to library and wait for copy of book

user can login/register and order particular book by filling all the correct information.

As the Application has a GUI format, it is user friendly too. It is a unique experience developing my first project due to which we have gained a lot of knowledge and interest not only in Software Development but also in Project Management.