**E-BOOK REPOSITORY**

**PROJECT REPORT**

*Submitted by*

**DILEEP KUMAR (12BIF0044)**

**Faculty: Prof. Lydia Jane**

**Software Engineering**

**CSE325**



Contents

1. Introduction 3

1.1 Purpose 3

* 1. Scope 3
  2. Definitions, acronyms, and abbreviations 3

1. Planning and Scheduling 4
   1. SDLC Model 4
   2. Gantt Chart 5
   3. Pert Chart 6
2. Software Requirement Specification 6
   1. Functional Requirements 6
   2. System Requirement 6
   3. Non-Functional Requirements 7
   4. Assumption and Constraint 8
3. Software Design Specification 8
   1. ER Diagram 8
   2. Activity Diagram 9
   3. Class Diagram 10
   4. Data Flow Diagram 11
   5. State Diagram 12
   6. Sequence Diagram 13
   7. Architecture Diagram 13
   8. User Interface Diagram 14
4. Testing 15
   1. Test Case 15
   2. Final Test Case 15
   3. Test Execution Report 16
5. Conclusion 16
6. **Introduction**

With advances in technology and an increase in the quality of digital paper displays, which provide users with an experience that is similar to reading from actual paper pages, there is no doubt that eBook reading will continue to grow in popularity. As with most electronic devices, as popularity grows, technology continues to evolve and enhance the user experience. So we have provided an eBook portal for the customer from any place, so that they can access free eBooks at any time, without carrying a bunch of heavy books. The users have the following facilities that they can use just by singing up to our eBook website.

**1.1 Purpose**

Defining and describing the functions and specifications of the E-Book Repository (EBR) is the primary goal of this Software Requirements Specification (SRS). This Software Requirements Specification illustrates, in clear terms, the system’s primary uses and required functionality as specified by our customer. The intended audience of this document is our primary E-book Repository System customer: any individual user, any kind of organization that will require access to such documentation.

**1.2 Scope**

The software system being produced is called Book EBR. It is being produced for a customer interested in reading books via Internet. This system is largely cross-platform and is available to anyone using Smart phone or Laptop that has internet accessibility. The system will be run on a central server with each user having a remote user interface through a web browser to interact with it.

The EBR will allow any user to create an account to become a customer. The customer, through the process of account creation, will have the option to become a member of the site. The system will allow customers to browse, search, select, and add books to repository. Then, provided they can check out books in repository. The EBR also allows a admin to manage the repository with full create account, read books, upload and QForum functionality with regards to books in the system.

The EBR will have numerous constraints on what it can do. Every customer should have unique username. The system also will not allow users to retrieve passwords or edit their user details.

* 1. **Definitions, acronyms, and abbreviations**

|  |  |
| --- | --- |
| EBR | E-BOOK REPOSITORY |
| Book | An instance of an Item that has these additional attributes: Title, Author |
| Button | A user interface element that allows a User to click and inform the |
|  | system to take an action |
| Checkbox | A user interface element that allows a User to inform the system that |
|  | he/she selected a particular item |
| Checkout | The process a Customer goes through to purchase an Item |
| VRUA | View, Retrieve, Update, Ask |
| Customer | A person that is a user of the system but has created an account |
| Item | An individual entity in the repository which has several descriptive |
|  | attribute |
| Member | A person that is a customer of the system and has proper account |
| SRS | Software Requirements Specification |
| Text Box | A user interface element that allows a User to input text to the system |
| User | The person who operate the software product. |

1. **Planning and Scheduling**

**2.1 SDLC Model**

**INCREMENTAL MODEL**

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle.



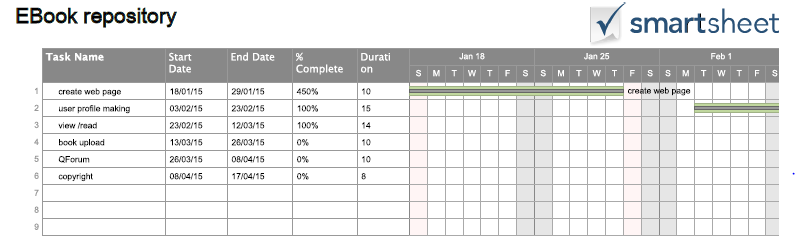
Why Incremental Model?

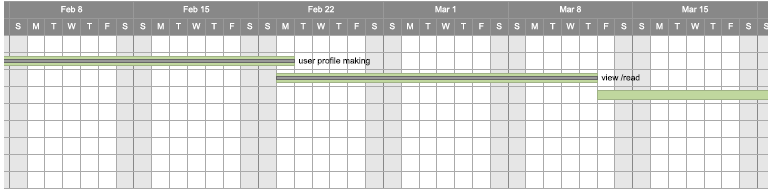
1) All the major requirements are well defined however some of them might evolve over the time hence incremental model provides the scope to do so. Waterfall Model has less flexibility as compare to incremental.

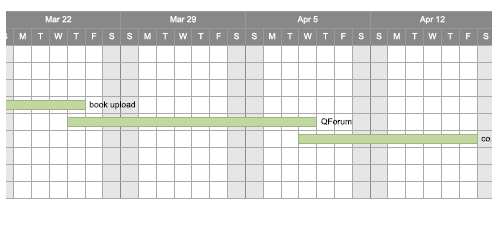
2) Testing and debugging can be done easily at every iteration hence making it easy to handle risks.

3) Also some new technology and tools are being used and hence it is more convenient to use incremental model as at every iteration we can review whether the tool used is giving the correct output as per the requirements and the platform on which it is being built.

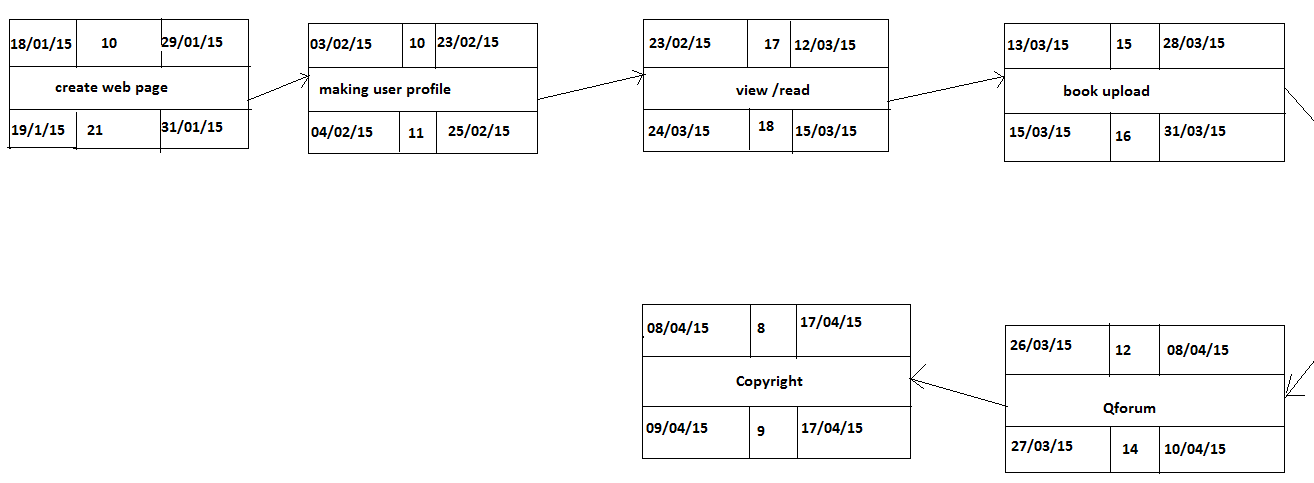
2.2 Gantt chart







2.3 Pert Chart



1. **Software Requirement Specification**
   1. Functional Requirements
2. To facilitate members to form group’s to organizational local meetings, e-book question forum’s to discuss the view among the different members.
3. To provide space for registered users to post about the interesting books which are not available at servers.
4. Secure registrations for new users.
5. To provide users to search various genres of interesting books.
6. There will be a module to check the copyright related issues.
   1. **System Requirements**

**HARDWARE INTERFACE**

**Client Side:**

**Browser-** Internet Explorer - 6

**Processor-** Intel Pentium III or AMD -800 MHz

**RAM-** 128 MB

**Disk Space-** 100 MB

**Server Side:**

**Browser-** Internet Explorer - 6

**Processor-** Intel Pentium III or AMD -800 MHz

**RAM-** 1 GB

**Disk Space-** 3.5 GB

**SOFTWARE INTERFACE**

**Client on Internet**

Web Browser- Chrome, Mozilla, Internet Explorer, Operating System- Windows 7, 8

**Client on Intranet**

Web Browser- Chrome, Mozilla, Internet Explorer, Operating System- Windows 7, 8

**Data Base Server**

MYSQL, Operating System -Windows 7, 8

**Development End**

PHP, HTML, AJAX, JAVASCRIPT, CSS, MYSQL, OS (Windows)

**3.3 Non-Functional Requirements**

**1) Performance**

**Response Time :-** The system shall give responses in 1 second after checking the patients information.

**Capacity:-** The System must support 1000 people at a time.

**User- interfac :-** The user-interface screen shall respond within 5 seconds.

**Conformity:-** The systems must conform to the Microsoft Accessibility

**2) Security**

**Customer Identification:-** The system requires the customer to identify himself /herself using Username and password.

**Logon-ID:** Any user who uses the system shall have a Logon ID and Password.

**Modification:** Any modification (inert, delete, update) for the Database shall be synchronized and only by the administrator in the ward.

**Administrators**' **Rights:-** Administrators shall be able to view and modify all information in EBook Repository Data.

**3) Reliability**

How general the form generation language is Simplicity vs. functionality of the form language= Speeds up form development but does not limit functional.

**4) Availability**

The system shall be available all the time.

**5) Safety**

Humans are error-prone, but the negative effects of common errors should be limited. E.g., users should realize that a given command will delete data, and be asked to confirm their intent or have the option to undo.

**6) Software Quality**

Good quality of the framework= produces robust, bug free software which contains all necessary requirements Customer satisfaction.

**7) Reusability**

Is part of the code going to be used elsewhere= produces simple and independent code modules that can be reused

**8) Maintainability**

**Backup:** The system shall provide the capability to back-up the Data.

**Errors:** The system shall keep a log of all the errors.

* 1. **Assumptions & Constraints**

1. Login and password is used for the identification of users.

2. Only registered customer will be authorized to use the services.

3. Limited to HTTP/HTTPS.

4. This system is working for single server.

5. All the members will be having of unique id and combination of username and password.

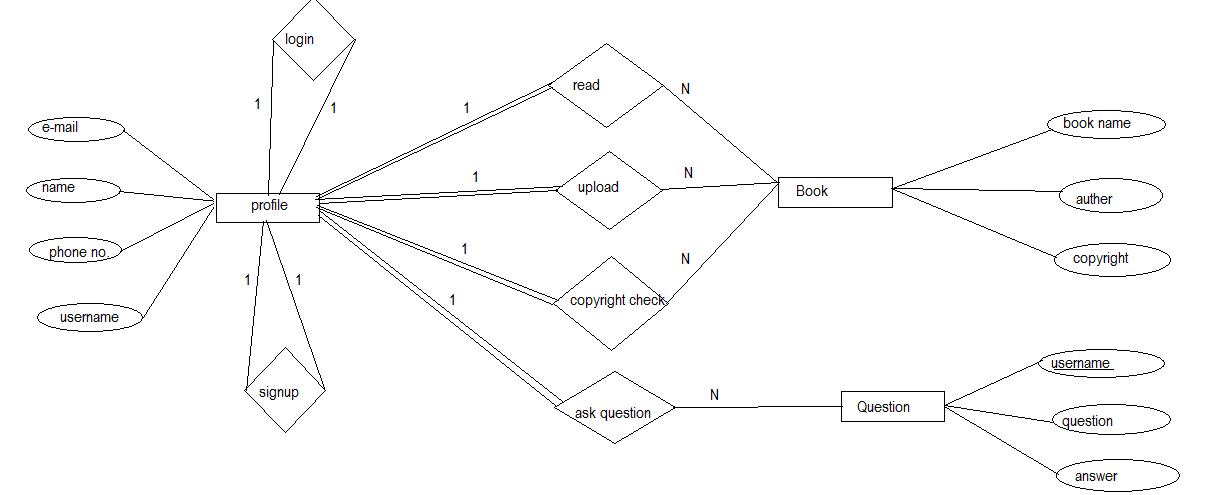
1. Software Design Specification

4.1.Entity Relationship

# Introduction

In software engineering, an entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

An entity–relationship model is a systematic way of describing and defining a business process. The process is modeled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

Diagram:

4.2.Activity Diagram

# Introduction

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

Rounded Rectangles=>Actions;

Diamonds => decisions;

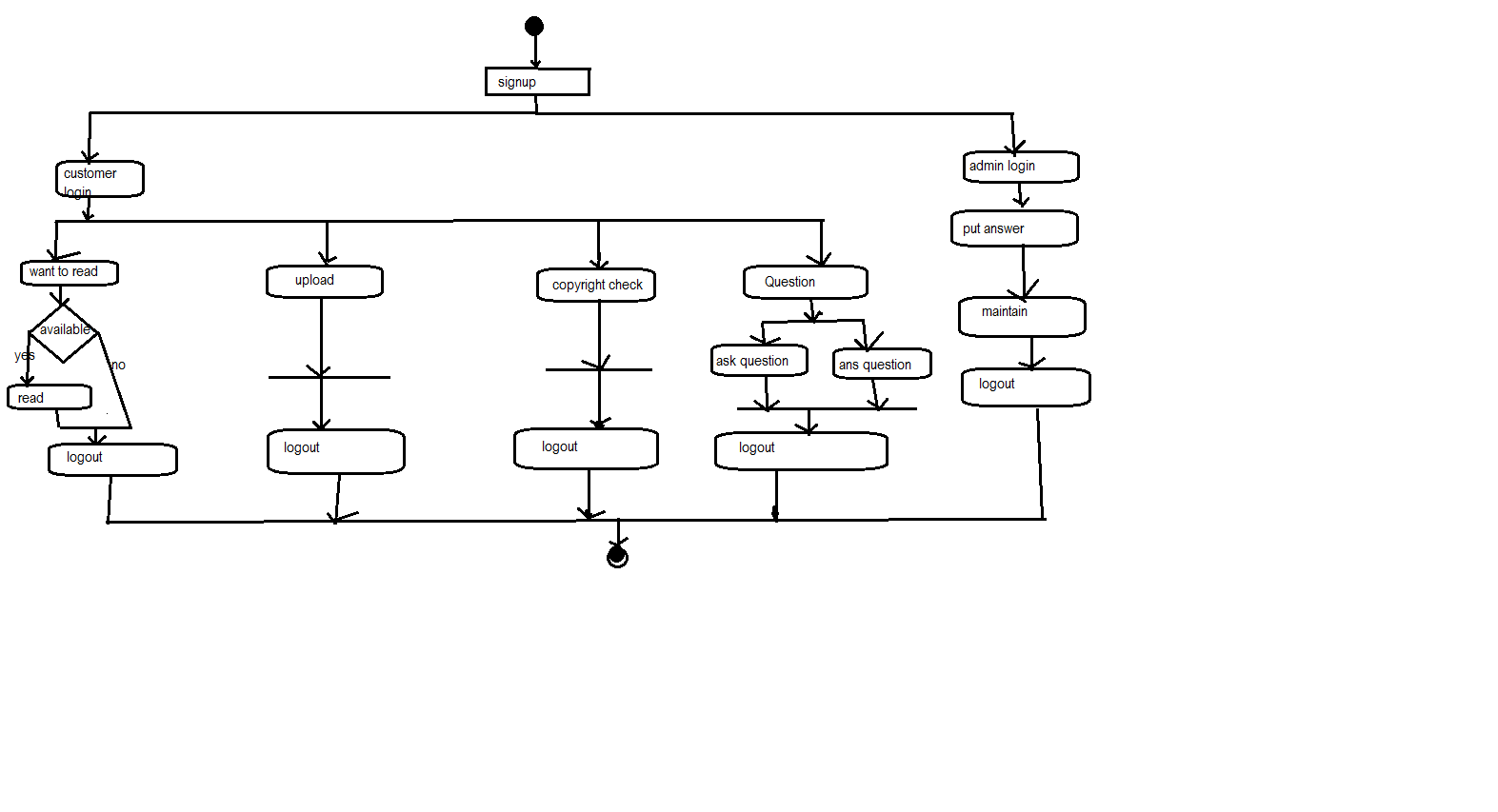
Bars=> start (split) and end (join) of concurrent activities;

Black circle=>initial state of the work flow;

Encircled black circle=>end (final state);

Arrows run from the start towards the end and represent the order in which activities happen. Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency [citation needed]. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

Diagram:



4.3. Class Diagram

# Introduction

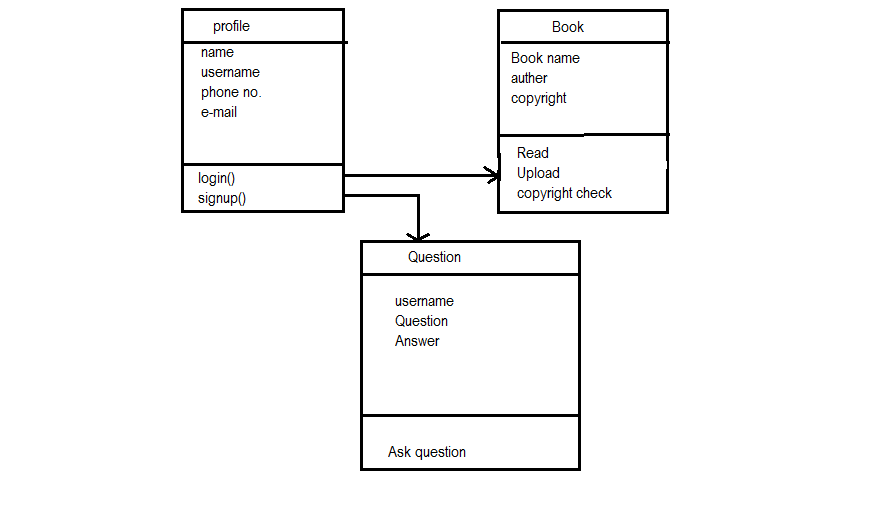
The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, inter-actions in the application and the classes to be programmed. A class has three sections.

In the diagram, classes are represented with boxes which contain three parts:

1. The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
2. The middle part contains the attributes of the class. They are left-aligned and the first is lowercase.
3. The bottom part contains the methods, the class can execute. They are also left-aligned and the first letter is lower case.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

Diagram:



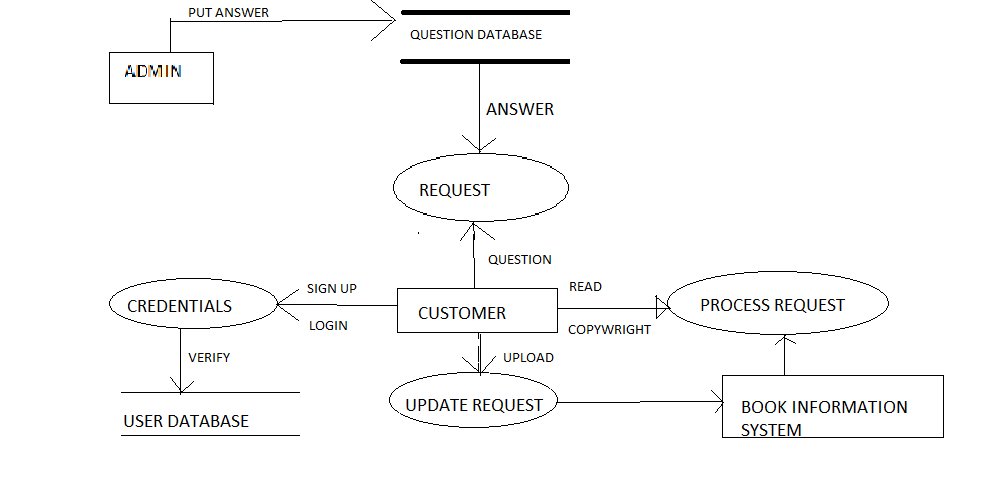
4.4. Data Flow Diagram

# Introduction

A Data Flow Diagram (DFD) is a graphical representation of the ”flow” of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

# Diagram

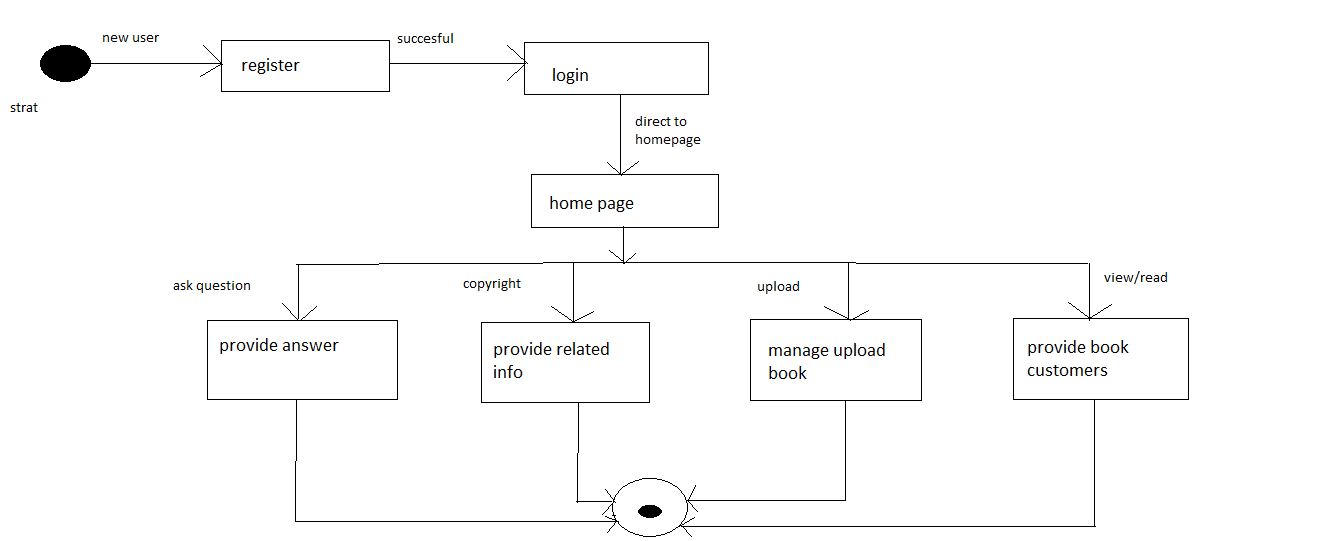


**4.5. State diagram**

# Introduction

State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and rep-resented in series of events, which could occur in one or more possible states. Hereby” each diagram usually represents objects of a single class and tracks the different states of its objects through the system”.

Diagram:



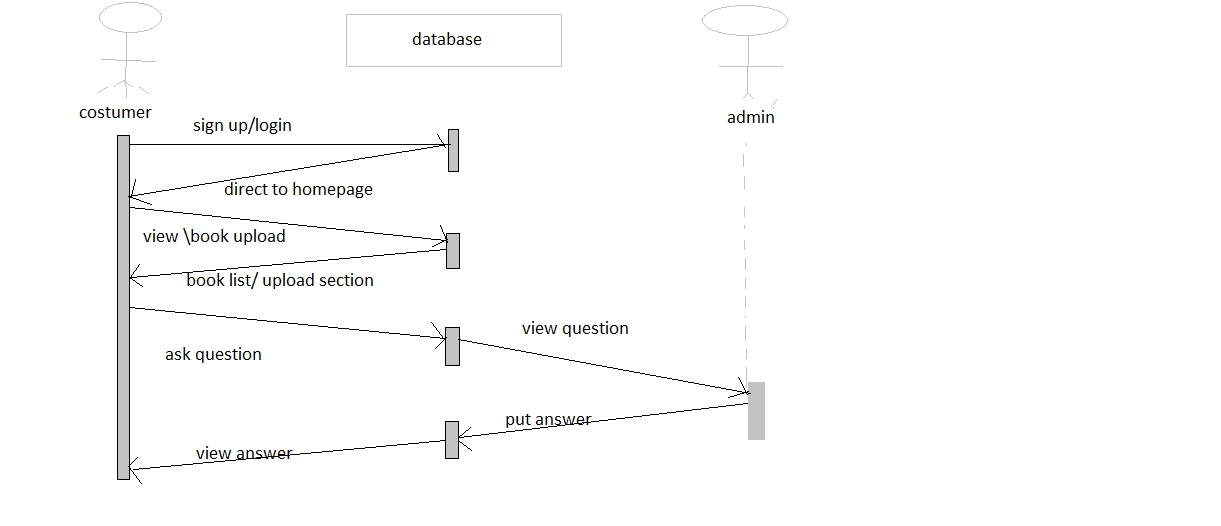
**4.6. Sequence diagram**

# Introduction

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. 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 are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Diagram:



* 1. **Architecture Diagram**

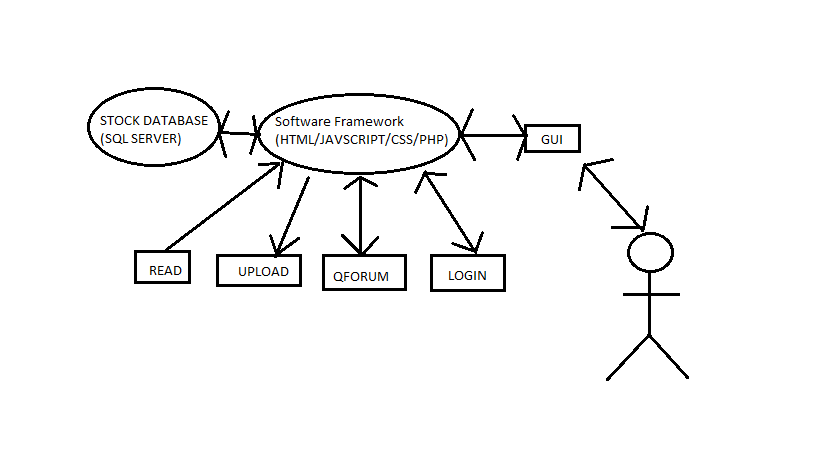
**Introduction:**

Software architecture is the high level structure of Software system, the discipline of creating such structures, and the documentation of these structures. It is the set of structures needed to reason about the software system, and comprises the software elements, the relations between them, and the properties of both elements and relations.

**It’s characteristics are:**

1. Multitude of Stakeholders
2. Separation of Concerns
3. Quality-driven
4. Recurring styles
5. Conceptual integrity

Diagram:

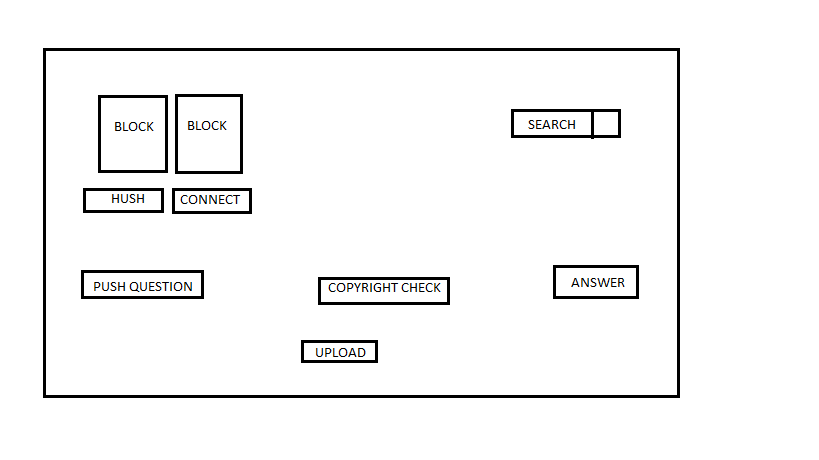


* 1. **User Interface Diagram**

**Introduction:**

Specifying user interfaces consists in a fundamental activity in the user interface development life cycle as it informs the subsequent steps. Good quality specifications could lead to user interface that satisfies the user’s needs. The user interface development life cycle typically involves multiple actors possessing all their own particular inputs of user interface artifacts expressed with their own formats, thus posing new constraints for integrating them into comprehensive and consistent specifications of a future user interface.

Diagram:



1. Testing

**5.1. Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Test Case | Exp.\_Result | Act.\_Result | Test\_outcome |
| 1 | Selects sample books available at website | Book will be open in separate tab | Book opened  In separate tab | Passed |
| 2 | User puts  Question | Question will post successfully | Error in posting | Failed |
| 3 | User checks for copyright | Shows copyright result | Error in checking | Failed |
| 4 | User upload the book | Book will be uploaded | Error in uploading | Failed |
| 5 | User is able to login properly | Redirect to home page | Redirected to home page | passed |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5.2. Final Test Case:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | S.NO | Test Case | Exp.\_Result | Act.\_Result | Test\_outcome | | 1 | Selects sample books available at website | Book will be open in separate tab | Book opened in separate tab |  | | 2 | User puts  Question | Question will post successfully | Question posted successfully | Failed | | 3 | User checks for copyright | Shows copyright result | copyright result | Failed | | 4 | User upload the book | Book will be uploaded | Uploaded | Failed | | 5 | User is able to login properly | Redirect to home page | Redirected to home page | passed |  * 1. **Test Execution Report:**   **Iteration/ Cycle** | | | |
|  | | | |
|  | | | |
|  | | | |
| *EXECUTED* | PASSED | | 2 |
| FAILED | | 3 | |
| *(Total) TESTS EXECUTED*  *(PASSED + FAILED)* | | 5 | |
| PENDING | | 0 | |
| IN PROGRESS | | 0 | |
| BLOCKED | | 0 | |
| *(Sub-Total) TEST PLANNED* | | **5** | |
| *(PENDING + IN PROGRESS +*  *BLOCKED + TEST EXECUTED)* | | | |
|  | |  | |

**6. Conclusion**

This EBook Repository is a better option if it comes into use. It helps the user to get rid of carrying books and it also saves the wastage of papers up to larger extent. User can also avail this facility at very cheaper rates. Proper maintenance should be done in order to ensure proper working of website. The customer found this EBook portal very satisfiable.