

GROW TOGETHER

A

Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

Eesha Sarang Gandhi <1602-18-737-067>

J Gayatri <1602-18-737-068>

Under the guidance of

Ms S. Sree Lakshmi



Department of Information Technology
Vasavi College of Engineering (Autonomous)
(Affiliated to Osmania University)
Ibrahimbagh, Hyderabad-31

ABSTRACT

Grow Together digitalizes the conventional library system and is easily accessible to anyone. It aids everyone in being beyond the usual. A simpler and affordable form of kindle with many more functionalities. The thought of having a system that could help the society in a different way and aid in self-growth triggered the idea for this software development activity.

It provides various functionalities such as selling, buying, renting and borrowing of books. In addition to these, it also serves as a platform to raise awareness and funds in case of emergencies, to start and be a part of book clubs and most importantly offers a space for reviewing and rating books.

Upon becoming a member the customer is provided with various functionalities as mentioned above. The clerks of this system keep an account of the status of the borrowed books and maintain a record of the funds. The sellers can add books along with raising awareness and funds. One day before a book is due to be returned, an email is sent to the customer.

Reliable and flexible, Grow Together is your one-stop destination for exploring a whole new world of knowledge and your inner love for reading.

PROBLEM ANALYSIS

Existing System

Online libraries such as amazon kindle unlimited, wattpad are platforms that already exist and enable the user to pay a certain amount for a month to provide unlimited access to a vast library of books. The main notion behind these platforms is to provide a space for book worms and reading enthusiasts to read their favorite books on the go, from the comfort of their own device. These platforms are quite popular and have seen tremendous growth since their initialization due to the sheer comfort they provide the user with. Some of the functionalities included in the amazon kindle unlimited system are, users own personal library, allowing the user to buy books, allowing the user to partner with the brand to sell their books. Additional features included in the wattpad system along with the above mentioned are, ability to meet and chat with other reading enthusiasts, ability to sell your book at your desired price without the need for support of publishers.

Proposed System

Grow Together digitalizes the conventional library system so that it is easily accessible to everyone. It can be viewed as a simpler and affordable form of kindle unlimited and wattpad mentioned above with additional functionalities. Where a conventional library would just allow you to borrow books and journals, Grow Together offers you much more than that. It has various functionalities like selling, buying, renting and borrowing a book similar to the online libraries available but in a simpler form. Apart from this, it has two special features- raise funds and raise awareness. These functionalities let the users to raise funds during any emergency situation be it a catastrophe or a medical urgency and spread awareness about the topics that they think need people's attention. The raise awareness functionality was introduced to mimic the concept of a newspaper. The main aim behind Grow Together was to provide an all-in-one space for the user.

SOFTWARE REQUIREMENTS SPECIFICATION

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References	1
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment	2
2.5 Design and Implementation Constraints	2
2.6 User Documentation	2
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	3
3.1 User Interfaces	3
3.2 Hardware Interfaces	3
3.3 Software Interfaces	3
3.4 Communications Interfaces	3
4. System Features	4
4.1 System Feature 1	4
4.2 System Feature 2 (and so on)	4
5. Other Nonfunctional Requirements	4
5.1 Performance Requirements	4
5.2 Safety Requirements	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
5.5 Business Rules	5
6. Other Requirements	5

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification document is to clearly define the initiative- Grow Together.

Grow Together digitalizes the conventional library system and is easily accessible by anyone. It aids everyone in being beyond the usual. A simpler and affordable form of kindle with many more functionalities.

1.2 Document Conventions

	Font	Style	Size
Heading	Times New Roman	Bold	16
Subheading	Times New Roman	Bold	16
Body	Times New Roman	Plain	12

1.3 Intended Audience and Reading Suggestions

The intended audience of this document includes the book owners, the fundraisers and the end readers. Other intended audience includes the development team such as the requirements team, requirements analysts, design team, and other members of the developing organization. The document can be read by any person having basic knowledge about software engineering. It can be easily understood by an amateur in this category as the document is written in simple English.

1.4 Product Scope

The thought of having a system that could help the society in a different way and aid in self-growth triggered the request for this software development activity. It provides various functionalities such as selling, buying, renting and borrowing of books. In addition to these, it also serves as a platform to raise awareness and funds in case of emergencies, to start and be a part of book clubs and most importantly offers a space for reviewing and rating books.

1.5 References

Fundamentals of python 3 and database management systems.

2. Overall Description

2.1 Product Perspective

Grow together is a web-based platform which revamps the conventional library system by increasing the functionality and the number of modules. The system provides a secure environment for all funds raised by the members and for the storing and retrieving of confidential data.

2.2 Product Functions

Grow together allows its customers to buy, sell and rent books online. In addition to this, it also raises awareness about important affairs and helps in raising funds for social causes. To access these benefits one needs to sign up for the membership plan. Upon becoming a member the customer is provided with various functionalities as mentioned above. The clerks of this system keep an account of the status of the borrowed books and maintain a record of the funds. The sellers can add books along with raising awareness and funds. One day before a book is due to be returned, an email is sent to the customer.

2.3 User Classes and Characteristics

This software can be used by people to sell, rent, buy or borrow books. Thus anyone who has ebooks can put their ebooks on this app for other users to rent, buy or borrow, depending on the mode the owner of the book offers. Users interested to raise money for a social cause can offer books, start a book club or offer services for patrons which will help them raise money.

2.4 Operating Environment

Operating environment for this system is as listed below.

- client/server system
- Operating system: Windows
- database: MySQL
- platform: java/python

2.5 Design and Implementation Constraints

This system provides web access for all its functionalities. The user interface will be intuitive enough so that no training is required by any customers, members or administrators. All online financial transactions and the storage of confidential member information will be done in a secure environment. Persistent storage for membership, funds and information about any book will be maintained.

2.6 User Documentation

Though the software will be extremely user-friendly, a manual containing the major functionalities will be provided to the user. Along with this, a tutorial regarding the usage will be included to make it simpler for the user to understand and handle it.

2.7 Assumptions and Dependencies

The users should have sufficient knowledge about computers and make sure that their system is connected to the internet. The users should know the English language as the user interface will be provided in English.

3. External Interface Requirements

3.1 User Interfaces

- When the user will open the software a welcome window will appear.
- In the login window, the user can easily enter the desired password and login name.
- From every window, the user can easily go to any desired window that is there will be absolute and relative linking.
- There is help and support option is present for the ease of the user.
- There will be a proper collection of GUI interface, which will provide a better look and feel.
- In the screen layout, the background colour will be very light and the graphics and font style will be in proper manner and well organized.
- If the user commits an error while typing, a proper error message will be displayed.
- In each and every window there will be alert, confirm etc message box for displaying the message.
- The user will be able to search for any data from the record by using proper guidelines.
- The opening page of the software will contain a menu window where the overall table contents of the software will be present through which the user can move to any desired

window & Mac. This will provide better security data because the menu window will be displaying according to the login (admin or normal user).

- User can easily save its data into the database and keep track of the records of purchase, vendor and inventory etc.

3.2 Hardware Interfaces

- Operating system: window
- Hard disk:40 GB
- RAM: 256 MB
- Processor: Pentium(R)Dual-core CPU

3.3 Software Interfaces

A system with windows operating system and browser that supports DBMS.

3.4 Communications Interface

The online library system will be connected to the World Wide Web.

4. System Features

4.1 Registration

4.1.1 Description and Priority

Registration is required by all users. It is to add the user and their details to the software's database which makes it easier for the user to make use of all the features. This has a priority of 10 on a scale of 0 to 10 since every user is required to be registered in order to access the features of the software.

4.1.2 Stimulus/Response Sequences

The user in order to register must follow the following steps,

- Provide their email address
- Provide their name, age and date of birth.
- Choose a username
- Set up a password that preferably follows all the security conventions.

4.1.3 Functional Requirements

The software should respond quickly to the user. Verify if the user has previously registered or is a new user. If the user is a new user, the software will have to show the user, a page to register and show a

message if any of the information entered doesn't follow all the rules and conventions.

4.2 Rent/Borrow/Buy

4.2.1 Description and Priority

This feature of the software allows users to rent, borrow or buy books provided by other users. Since it is the basic functionality of any library, the priority of this feature is at an 8 on a scale from 0 to 10.

4.2.2 Stimulus / Response Sequences

- User Registration or
- Login: for an already existing/ registered user
- Browse: enables the user to search through the database to find the book they prefer.
- Rent: once, the user finds a book they like, they can rent it for a selected duration of time(choice of the user renting the book), pay the amount of money to the buyer through Gpay or Paypal account and add the book to their library.
- Borrow: After selecting a book of their choice, the user can directly add the book to their library upon selection of a duration of time
- Buy: To buy a book choose on the buy option and transfer money through your Gpay or Paypal accounts.

4.2.3 Functional Requirements

The functionality of the software is to provide the user with the options The user has when they pick a book. The options can include, providing the user with a list of the durations for renting and borrowing the book and the cost for it if mentioned by the seller.

4.3 Sell

4.3.1 Description and Priority

This feature allows the user to put their ebooks for sale and earn a share of money from them. The priority of this is 8 too.

4.3.2 Stimulus/ Response Sequences

- Select the sell tab and set the price of the book.
- Connect your Gpay or Paypal account to your grow together account.

4.3.3 Functional Requirements

The functionality of this feature is to allow the user to sell their ebooks.

This feature provides the user with information on how to add a book onto the site and make it visible to other users to buy, rent or borrow it. Displaying an error message if any errors show up or a request was failed to be processed, the selling feature of the software makes it easier for the user to sell.

4.4. Put

4.4.1 Description and Priority

This feature allows the user or the administrator to write reviews for books of their choice, to start a book club and to start a donation for raising money for any catastrophes. The priority for this feature is a little less in comparison to others.

4.4.2 Stimulus/ Response Sequences

- For starting a book club the user needs to go to the book club tab of the software
- Selecting the write option, the user will be able to write the concept of their book club and any other information they wish to provide.
- To write reviews the user needs to go to the browse tab, choose the book they want to review and can write their review on the thoughts tab provided just under the book.
- In order to start a donation, the user needs to go to the donation tab and start one.

4.4.3 Functional Requirements

This feature of the software allows the user to start a book club, start a donation for any catastrophes or any emergency situations and to write reviews for their desired book. The functionality of this feature apart from allowing the user to input information is to make sure all the events taking place with respect to this feature are performed without any disturbance and error, which if occurs the software displays a message.

4.5 Clerk

4.5.1 Description and Priority

The clerk functionality of this software aids in keeping an eye on the transactions and other exchanges and ensure their security and safety. The priority of this feature is high as this feature is essential for maintaining a balance throughout the software.

4.5.2 Stimulus/ Response Sequences

- Work as an interface between the explorer and the heritor ensure timely return of borrowed and rented books, ensure a secure transaction for paid books.
- Ensure that the money donated to the donations/ funds are transferred securely to the user who started it.
- Be available to help the users in case of any queries.

4.5.3 Functional Requirements

This feature works as an interface between the users and is required To be available to help the users in case of any queries.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The software can withstand any disasters like power failures.
- At every phase output of any phase will be the input of some other phase and this will ensure that it is reliable and accurate.
- At the time of login, the username and password entered will be matched to the information stored in the database, hence only authenticated users will gain access.
- There will be multiple ways of retrieving data in a short time span.
- The software will be well supported with other embedded software like Notepad.
- The overall performance of the software will be reliable and enable users to work efficiently.

5.2 Safety Requirement

The database might get crashed at any time due to virus or operating system failure. Therefore it is required to take the database backup so that the database is not lost. Proper UPS or inverter facility should be there in case of power supply failure.

5.3 Security Requirements

All the customer information is protected so that any system failure or malware will not get access to it.

- The system will use a secured database.
- Normal users can just read the information without modifying it except their personal details.
- Different types of users will have respective access constraints.
- Proper authentication will be granted.

- The accounts of the members and admin will be separate so that only the admin can access and update the database.

5.4 Software Quality Attributes

- Adaptability-This software is adaptable by any organization.
- Availability- It is an open-source project.
- Flexibility- The operation may be flexible and reports can be presented in many ways.
- Maintainability- After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
- Reliability-The performance of the software increases the reliability of the software.
- Reusability-The data and record that are saved in the database can be reused if needed.

5.5 Business Rules

- Only administrators are allowed to modify the database.
- Only supervisors might approve cash refunds.
- The sellers can only add more books and not make any other changes to the system.

6. Other Requirements

There are no other requirements.

PROJECT MANAGEMENT PLAN

1. Project Summary

1.1 Project Overview

Grow together is a software that digitalizes the conventional library system and is easily accessible to everyone. It aids everyone in being beyond the usual. Basically it is a simpler and affordable version of kindle with many more functionalities.

1.2 Project Scope

The project scope is primarily to provide functionalities like buying,selling,borrowing or renting books. In addition to this it also serves at a platform to raise awareness and funds in case of emergencies. Rating and reviewing books is out of the scope of the system.

1.3 Development Process

We follow the waterfall model of software development as it is much simpler and easier than other methods.

1.4 Effort, Schedule and Team

The team comprises of two people.

Total Effort: 5 hours/week

Project Duration: 3.5 months

1.5 Assumptions Made

No major assumptions beyond what is stated in the SRS.

(Person A: 067 Eesha,

Person B: 068 Gayatri)

2. Detailed Effort and Schedule

	Task	Estimated effort	Start date	End date	Person
1	System design	10	Jan 7	Jan 24	A,B
2	Detailed design	12	Feb 1	March 13	A,B
3	Coding input module	9	March 13	March 27	A
4	Coding scheduling module	10	April 1	April 15	A,B
5	Coding output module	6	April 17	April 22	A,B
6	Test planning	6	April 23	April 29	A,B
7	Testing and integration	8	April 30	May 7	A,B

8	Rework and final	5	May 8	May 15	A,B
---	------------------	---	-------	--------	-----

3. Team Organization

We have a small team of two people. Equal work is assigned to both the people.

4. Hardware and Software Resources Required

We require a workstation with Eclipse, SQL database and RSA.

5. Quality Plan

The testing of the quality of this project is divided into the following four sections and is done in a detailed and controlled manner.

SRS Review: The SRS was reviewed by both the members of the team and also by the mentor assisting the team.

Design Review: Every component of the design element was reviewed personally by the members and the mentor assisting the team.

Unit Testing: Each member tested different units of the project individually. This was reviewed by each other.

System Testing: The review of system testing will be done thoroughly by each member of the team and will be based on the system testing plan priorly made.

6. Risk Management Plan

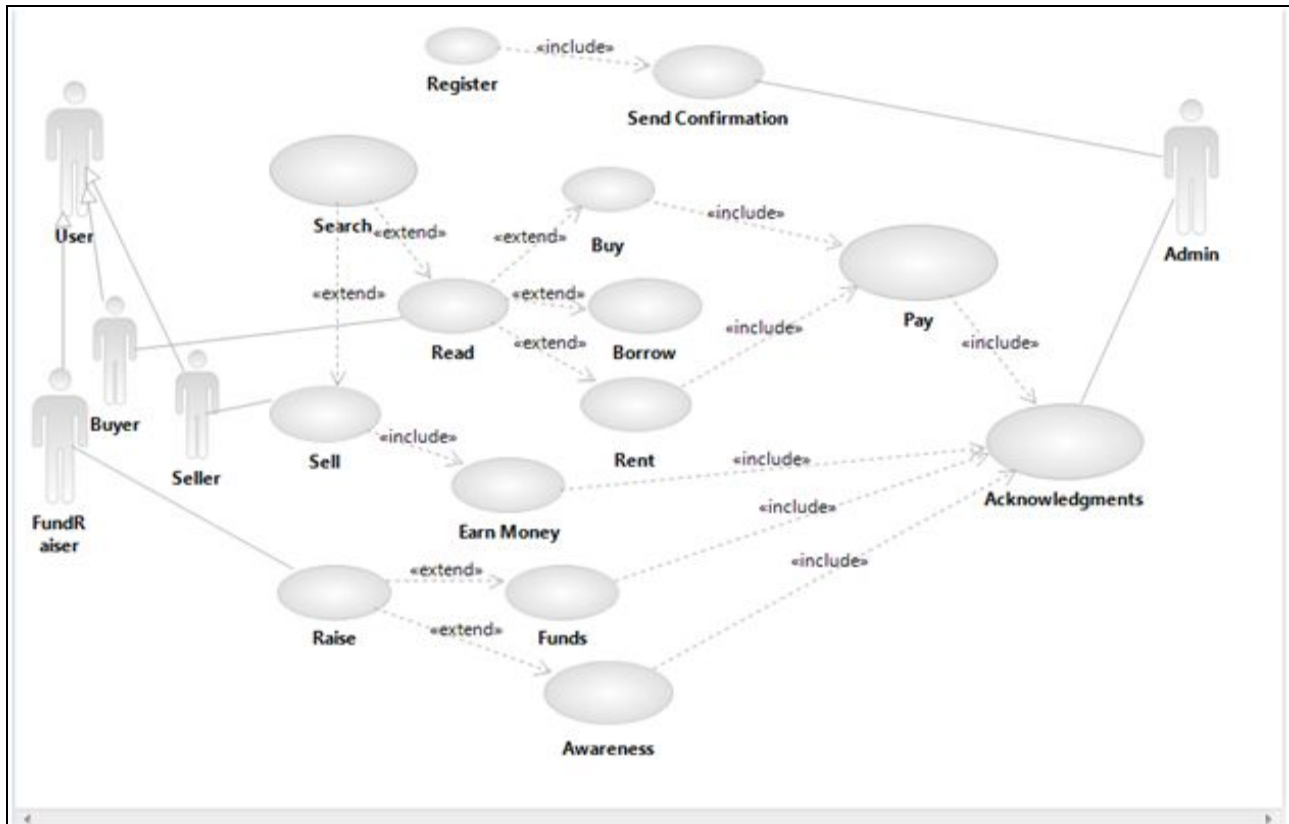
There are no major risks with this project that might need any explicit mitigation. Everything is more or less under control.

7. Project Tracking

Three basic methods will be used for monitoring – project logs, weekly meetings, and reviews. As there is no timesheet system, the project members will record their activity in a project notebook and report the hours for each activity in the meetings.

Reviews will be held as per the quality plan.

USE CASE DIAGRAM AND THE USE CASE SPECIFICATIONS



1. Register

Use case name: Register

Actor: User, Admin

Use case description: Allows the user to register themselves as a member in the Grow Together platform.

Pre condition: The user should have a valid email address to register as a member.

Post condition: The user should have confirmed their email address.

Main flow of events:

- The user starts registering by entering a valid email address.
- Then the user needs to fill in other details such as name, age and their username.
- Then the user needs to confirm their email address through a mail they will receive.
- Then, the member can start accessing the Grow Together platform.

Exceptional flow of events:

- The user enters invalid credentials such as an invalid email id, then they will have to try again with a valid email address.
- The user has not received a confirmation email and in such a situation, the user will have to try again with a valid address or try refreshing the website.

Special requirements: A log with the registered user's information should be maintained.

Extension points: The user can also register using their phone number.

Assumptions and dependencies: The user has a functioning email id and knows how to confirm it. The user is above the age of fourteen years.

2. Search

Use case name: Search

Actor: Buyer, Seller

Use case description: Allows the member to search for books or articles available on the platform.

Pre condition: The member should login with appropriate credentials.

Post condition: The member is able to view the book or article they were searching for.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by system.
- The member enters the name of the book/article they are searching for in the search bar.
- The book or article if available is displayed to the user.
- The book or article can be viewed by the member now.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again..
- The required book or article is unavailable. In this case the member can opt for a notification option so that they will be notified when it is available or they can retry searching with another name.

Special requirements: A log including the book name and user name should be maintained for notifying the user when the book is available.

Extension points: The member can also search for a book using the author's name.

Assumptions and dependencies: The person is already a member of Grow Together and has basic knowledge about how websites work.

3. Buy

Use case name: Buy

Actors: Buyer

Use case description: Allows the member to buy books or articles that are in stock.

Pre condition: The member should login with appropriate credentials.

Post condition: The member has received acknowledgement and has access to the book.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by system.
- The member searches for a book/article.
- The member places an order by making the payment.
- The member receives acknowledgement .
- The book can now be accessed by the member.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again..
- The required book is unavailable. In this case the member can opt for a notification option so that the book can be purchased when it is available.
- The card details are incorrect. The member is supposed to retry the payment.
- If there is insufficient balance in the account then the payment fails.

Special requirements: A log including the date and time of the purchase should be maintained.

Extension points: The member can rent a book for a specific period instead of making a purchase.

Assumptions and dependencies: The person is already a member of Grow Together and has basic knowledge about how websites work.

4. Pay

Use case name: Pay

Actors: Buyer

Use case description: Allows the member to pay for the books or articles that are in stock and that they want to buy.

Pre condition: The member should login with appropriate credentials.

Post condition: The member has received acknowledgement and has access to the book.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by system.
- The member searches for a book/article.
- The member places an order by making the payment.
- The member should enter valid card details and agree for processing the payment.

- The member receives acknowledgement once the payment is successful.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again.
- The book the member is searching for is not found, then he/she should try searching with a different name or can avail the notification option to receive an update as to when the book becomes available.
- The card details are incorrect. The member is supposed to retry the payment.
- If there is insufficient balance in the account then the payment fails.

Special requirements: A log including the date and time of the purchase should be maintained.

Extension points: The member can also pay using third party apps such as Gpay, Paytm and PhonePe.

Assumptions and dependencies: The person is already a member of Grow Together and has basic knowledge about how websites work.

5. Sell

Use case name: Sell

Actor: Seller

Use case description: Allows the member to sell books/articles.

Pre condition: The member should login with appropriate credentials.

Post condition: The article or book will appear in the display and the seller will get an acknowledgement regarding the same.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by the system.
- The member puts a book for sale along with its price.
- The member receives an acknowledgement.
- The book gets added to the display.
- The seller checks for orders.
- If any order is placed, the payment mode and the amount is verified.
- The book is sold.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again.
- If any book in an order is not available, a message regarding the same is sent to the buyer.
- If the payment is pending, the book will not be sold.

Special requirements: A log including the date and time of the purchase should be maintained and the number of copies should be updated.

Extension points: The seller can provide a renting option by mentioning the price for the specified period.

Assumptions and dependencies: The person is already a member of Grow Together and has a certain amount of professionalism.

6. Raise Awareness

Use case name: Raise awareness

Actors: Fundraiser

Use case description: Allows the member to provide a valuable piece of information regarding important events so that people are aware of what's really going around in the world.

Pre condition: The member should login with appropriate credentials.

Post condition: The article is verified and posted on the website.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by the system.
- The member uploads the article.
- The article is verified and made public.
- The article is now accessible to the readers.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again.
- The article does not pass the verification test. The article fails to be uploaded.

Special Requirements: The date and time of the upload should be recorded.

Extension points: None

Assumptions and dependencies: The person is already a member of Grow Together and has genuine news to share.

7. Raise Funds

Use case name: Raise funds

Actors: Fundraiser

Use case description: Allows the member to donate money for a social cause or in case of an emergency.

Pre condition: The person should login with appropriate credentials and browse through in order to identify where exactly the funds are needed.

Post condition: The amount will be used for charity.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by the system.
- The member goes through the list of charitable institutions that are associated with Grow Together.
- The member donates a certain amount to the selected institution or for the selected cause.
- The member receives an acknowledgement.

Exceptional flow of events:

- The member enters invalid credentials and hence he/she is supposed to login once again.
- The card details or the account details are invalid. The member is supposed to retry the payment.
- If there is insufficient balance in the account then the payment fails.

Special requirements: Minimum amount to be donated is Rs 500.

Extension points: None

Assumptions and dependencies: Information about charitable institutions and various other social causes which require funds is already provided.

8. Send Acknowledgment

Use case name: Send Acknowledgement

Actor: Admin

Use case description: Gives the member acknowledgement when payment is successful and any other jobs performed by the user that require acknowledgement.

Pre condition: The member should login with appropriate credentials.

Post condition: The member has received acknowledgement.

Main flow of events:

- The member enters the credentials in order to login.
- The credentials are verified by system.
- The member searches for a book/article.
- The member places an order by making the payment.
- The member receives acknowledgement .
- The book can now be accessed by the member.

Exceptional flow of events:

- The member enters invalid credentials, in this case he/she needs to login again.

- The required book is unavailable. In this case the member can opt for a notification option so that the book can be purchased when it is available.
- The card details are incorrect. The member is supposed to retry the payment.
- In case the acknowledgement isn't received, then the member needs to try refreshing the page, waiting for a few minutes and then check again or can contact customer care.
- If there is insufficient balance in the account then the payment fails.

Special requirements: The member needs to perform actions in order to receive acknowledgments. Such as after buying a book, the member will receive an acknowledgement of the purchase.

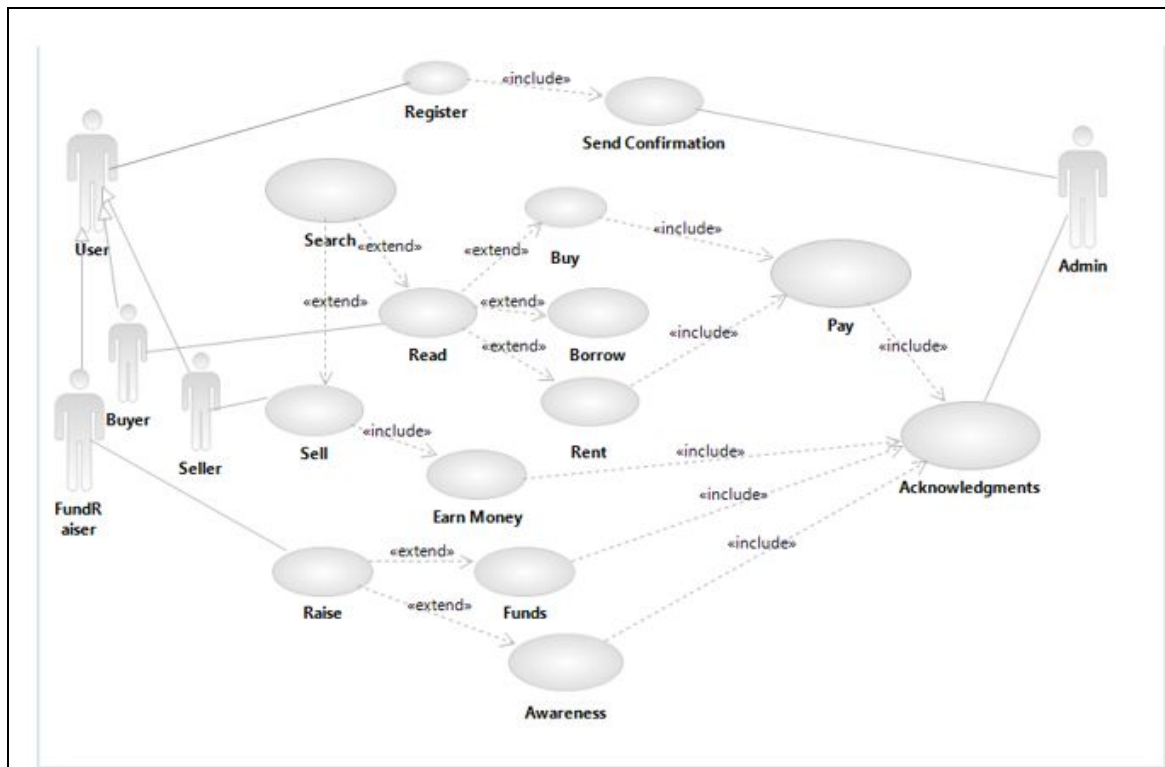
Extension points: The member can contact the customer care in case there is any delay or falsity in the acknowledgements received.

Assumptions and dependencies: The person is already a member of Grow Together and has basic knowledge about how websites work.

UML DIAGRAMS ALONG WITH DIAGRAM DESCRIPTIONS

1. Use Case Diagram

A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.



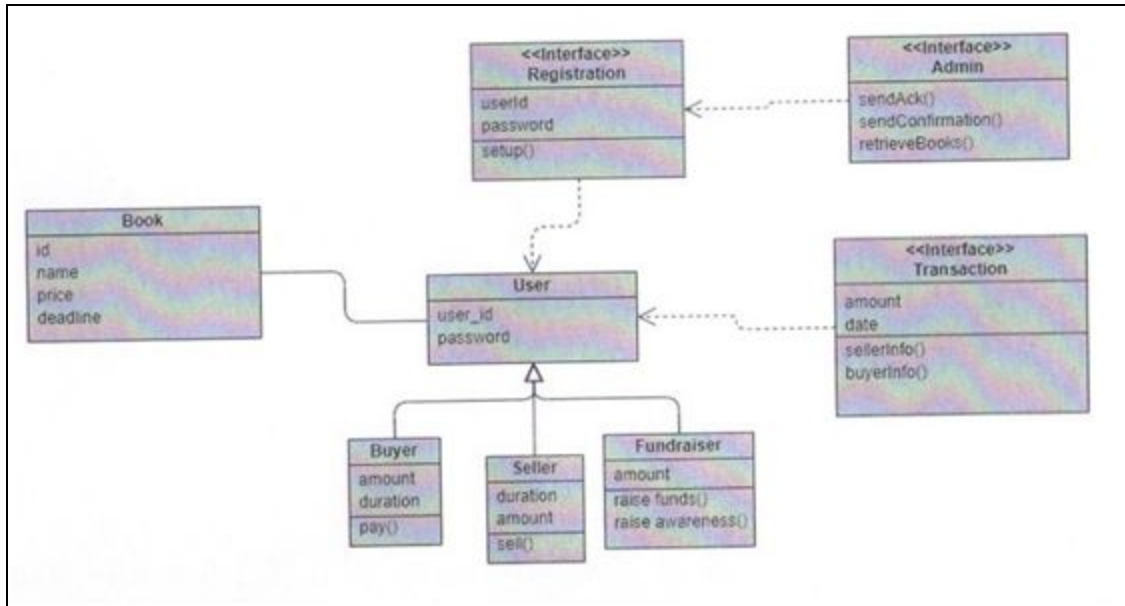
In the above use case diagram, the different types of actors are the admin and the users which are further generalised into the fundraiser, buyer and seller.

These actors are connected to the various use cases as depicted above.

The main use cases are register, search, buy, pay, sell, raise funds, raise awareness and send acknowledgements.

2. Class Diagram

A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

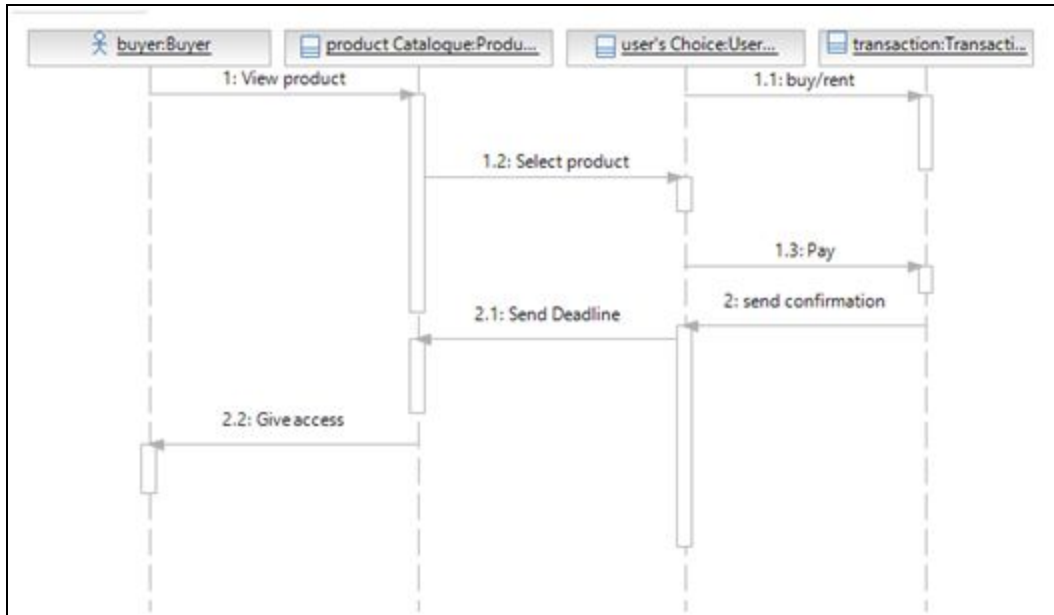


In the above diagram the user is generalized into three categories- buyer, seller and fundraiser.

The main classes are book and user and they implement three interfaces- registration, admin and transaction. There is a realization relationship between transaction and user, registration and user, admin and the user.

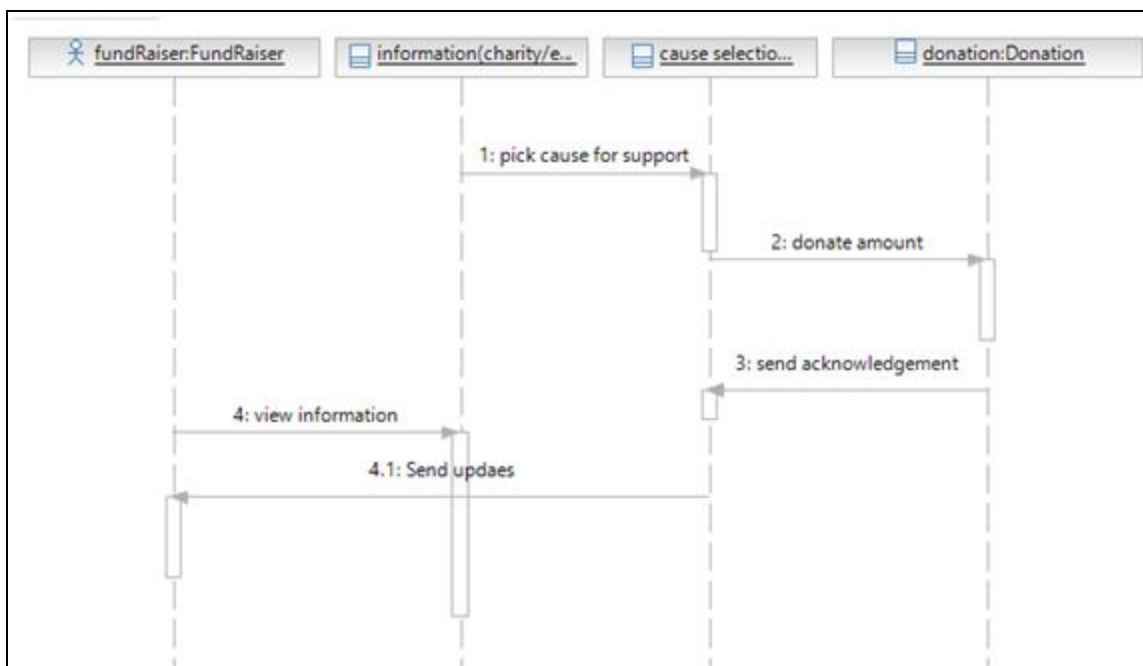
3. Sequence Diagram

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.



In the above diagram, the actor is the buyer and the identified objects include product catalogue, user's choice and transaction.

The buyer browses the product catalogue then selects the book and opts for either of the two options- buy or rent. After this the payment is made. The payment confirmation is sent along with the deadline if the member has opted for renting the book. After this the book can be accessed by the member.

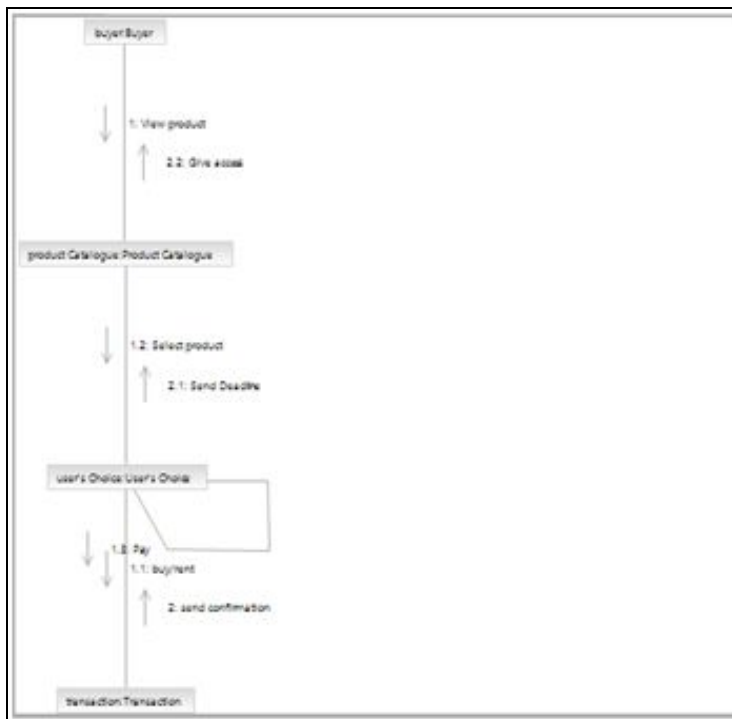


In the above diagram, the actor is the fundraiser and the identified objects include information, case selection and donation.

The member browses through the institutions and social causes which require funds. He/She then selects a cause and donates a certain amount. An acknowledgement regarding the donation is sent to the donor. The member can even get the information on how the funds are being utilized. Further updates are regularly sent to the member.

4. Collaboration diagram

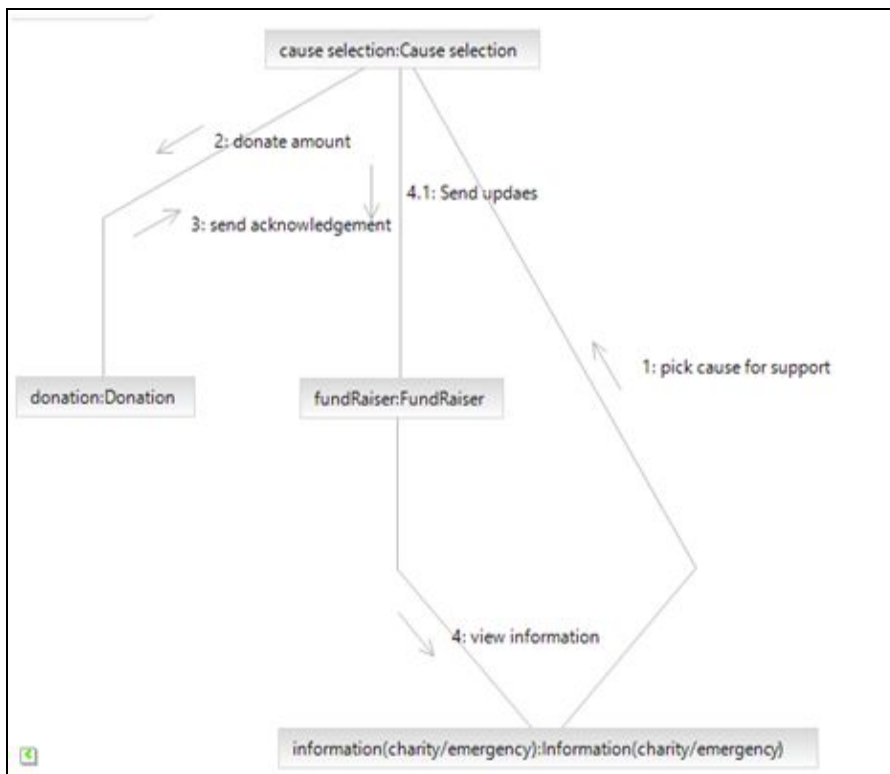
A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.



In the above diagram there are four objects- buyer, product catalogue, user's choice and transaction. These objects are connected with each other via links. A message on each link shows the interaction between objects.

The buyer browses the product catalogue then selects the book. After this the payment is made based on the selection of two options- buy or rent. The payment confirmation is sent

along with the deadline if the member has opted for renting the book. After this the book can be accessed by the member.

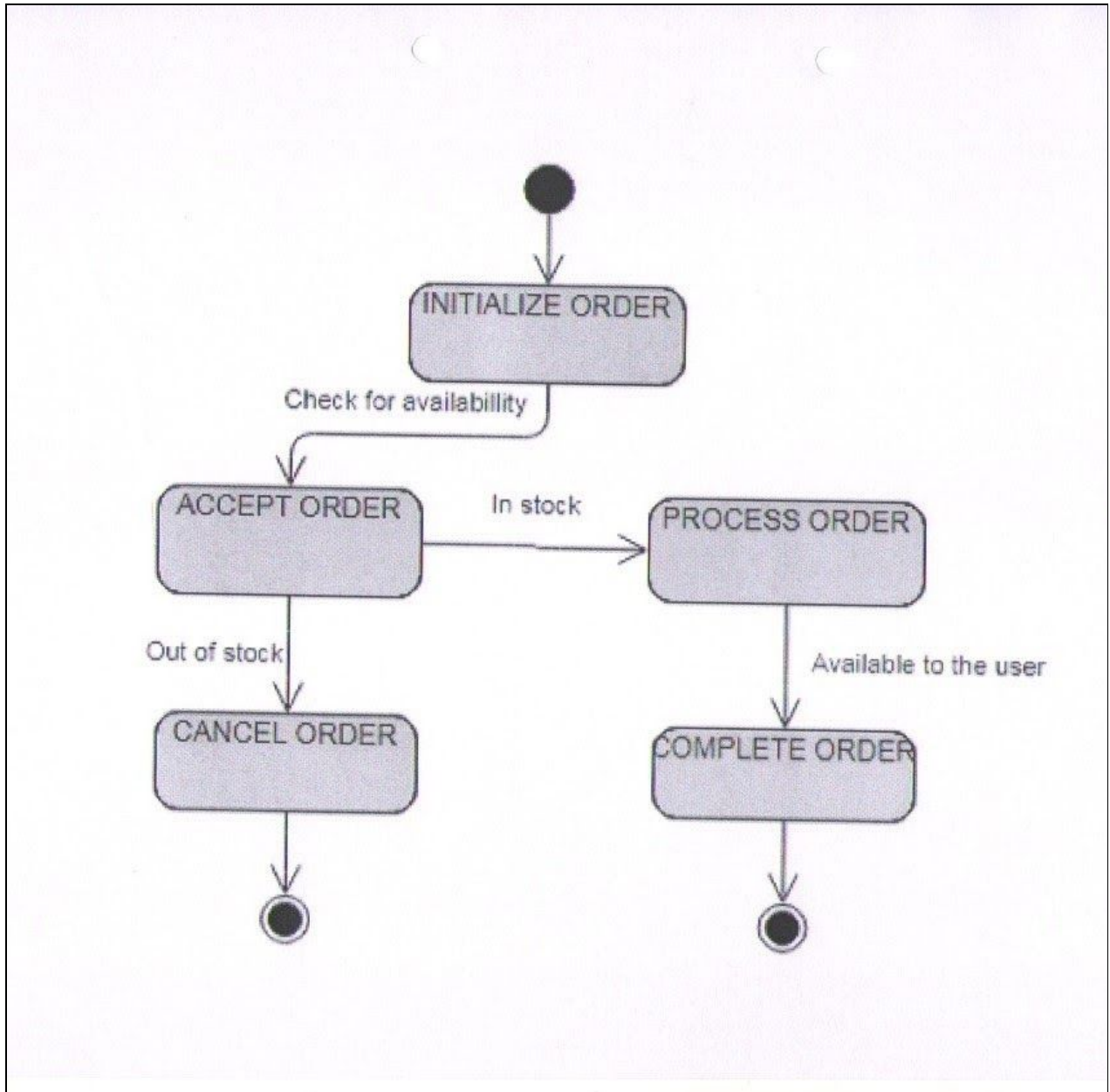


In the above diagram there are four objects- fundraiser, information, cause selection and donation. These objects are connected with each other via links. A message on each link shows the interaction between objects.

The member browses through the institutions and social causes which require funds. He/She then picks a cause and donates a certain amount. An acknowledgement regarding the donation is sent to the donor. Then the member gets the information on how the funds are being utilized. Further updates are sent to the member on a regular basis.

5. Statechart Diagram

The state chart diagram describes different states of a component in a system. The states are specific to a component/object of a system. This diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model the lifetime of an object from creation to termination.

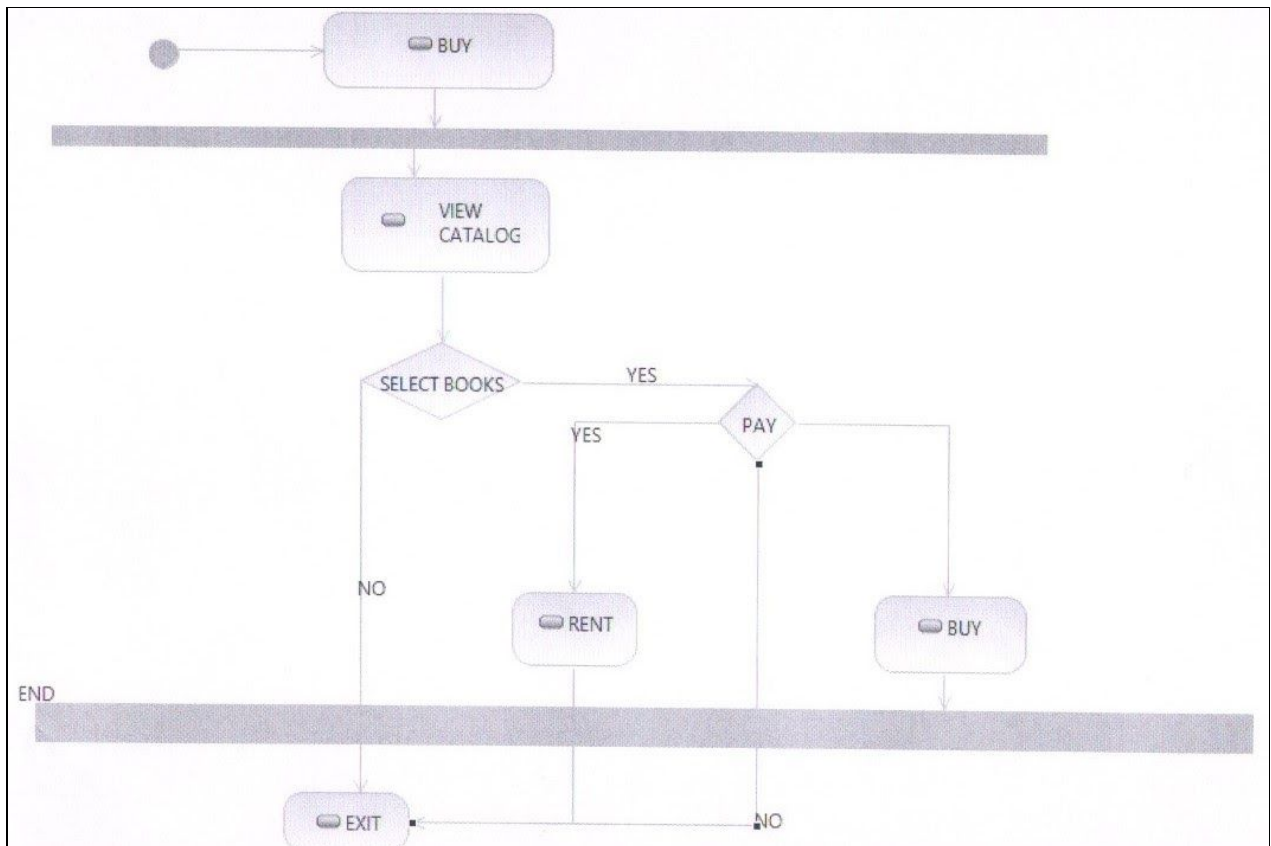


The above state chart diagram is for the object order. Since the initialization of the order until its termination, the states that the object undergoes is as depicted.

6. Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork and join.

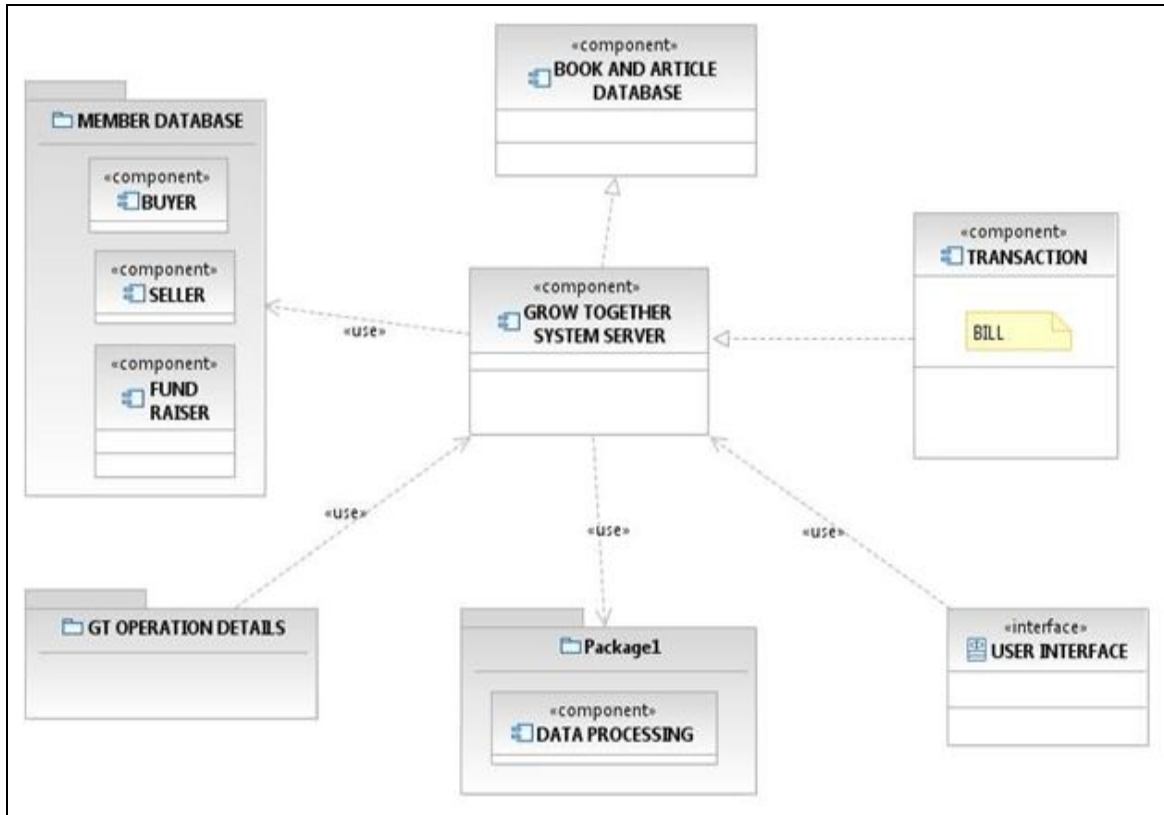


This activity diagram is shown for the use case buy. The sequence of operations that take place in this use case is as shown in the figure. The fork is used to depict the simultaneous operations that will be performed. The rhombus serves as a decision box, which takes a certain route based on the conditions given.

In this use case buy, the user can use the view the catalogue and select a book. he/she can then choose to buy the book and pay for it, or rent it or exit. These actions since they happen parallelly are under a fork.

7. Component Diagram

In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

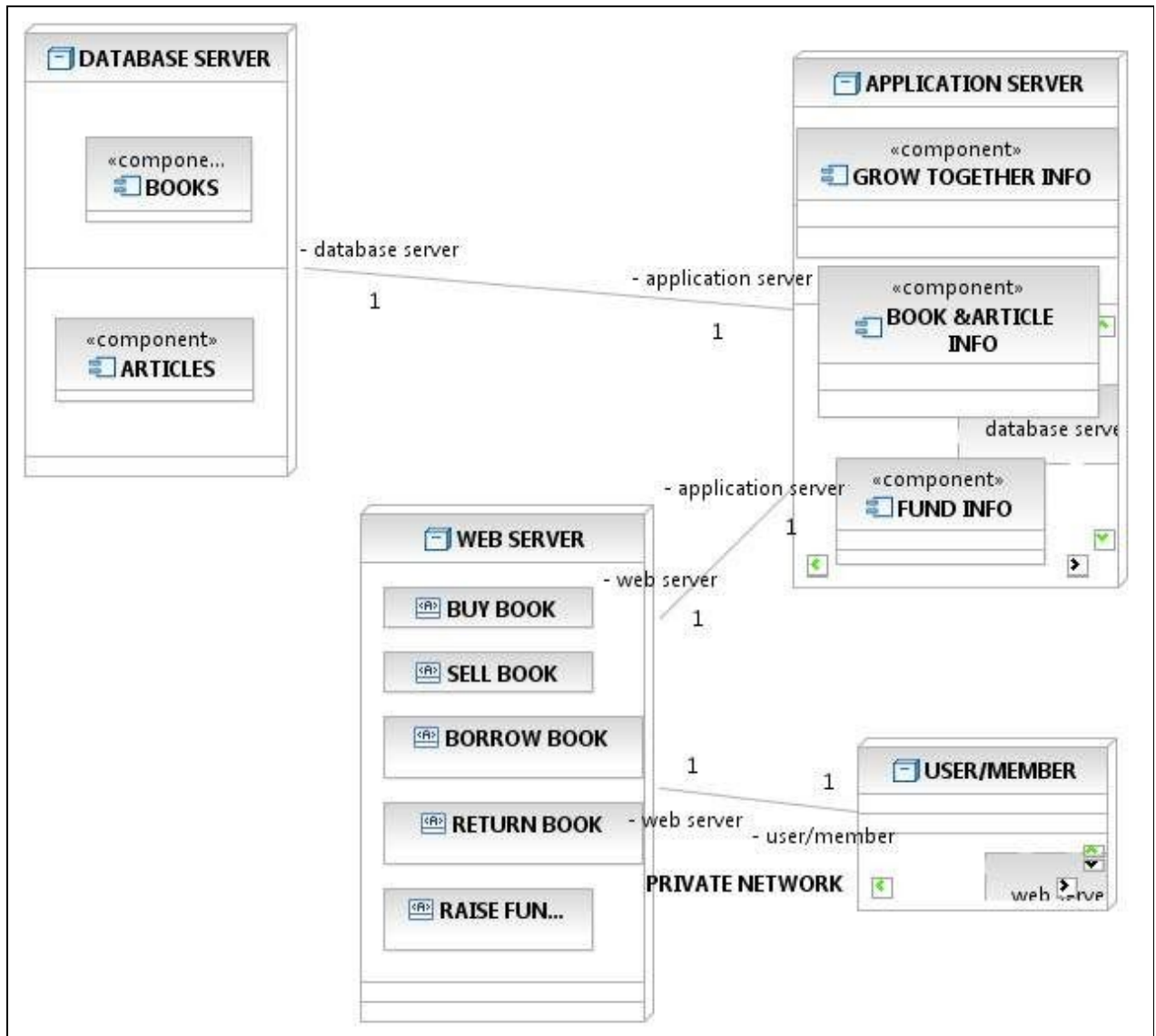


This diagram shows how the Grow Together system server uses other components and how it is related to them.

The user interface uses the server to load pages. The server accesses the book and article database to retrieve information from it. It also accesses the member database which includes information about all the users- the buyers, sellers and fundraisers. The data retrieved by the server from other components resides in the utility package where data processing will take place. Every component can get an access to the server only through GT operation details rather than accessing it directly.

8. Deployment Diagram

Deployment diagrams are used to describe the physical components (hardware), their distribution, and association. These diagrams can be visualized as the hardware components/nodes on which the software components reside.



The user accesses the web server through his/her private network. The webpages get downloaded on the user's web browser. The web server accesses the application server through the host address. According to the user's action the data about the sale of books, addition of funds etc gets stored onto the database.

CODING

Use case: Buy Books

```
import java.util.*;
import java.sql.*;
public class BuyBookTest
{
    static Connection connection;
    static Statement statement;
    public BuyBookTest()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and load driver");
            System.exit(1);
        }
    }

    public static String buybook(int buyer_id,int book_id,String when,float rate)
    {
        String check="Not Bought";
        try
        {
            connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","gayatri","manager");
            statement = connection.createStatement();
            String query1="select buyer_id,book_id,when,rate from buys";
            PreparedStatement ps=connection.prepareStatement(query1);
            ResultSet rs=ps.executeQuery();
            while(rs.next())
            {
                int bu=rs.getInt("buyer_id");
                int bo=rs.getInt("book_id");
                String w=rs.getString("when");
```

```

        float r=rs.getInt("rate");
        if(bu==buyer_id && bu==book_id && w.trim().equals(when) &&
r==rate)
        {
            check="Already Bought";
            break;
        }
    }
    if(check=="Already bought")
    {
        check="Book already bought, choose another one";
    }
    else
    {
        String query= "INSERT INTO buys
VALUES('"+buyer_id+"','"+book_id+"','"+when+"','"+rate+"')";
        int i = statement.executeUpdate(query);
        check="Successfully bought";
    }
}
catch(Exception e)
{
    check="Exception or Invalid input.Please try again!";
    System.out.println(e);
}
return check;
}
public static void main(String[] args)
{
    BuyBookTest b=new BuyBookTest();
    int buyer_id,book_id;
    String when;
    float rate;
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter buyer_id");
    buyer_id=sc.nextInt();
    System.out.print("Enter book_id");
    book_id=sc.nextInt();
    sc.nextLine();

```



```

        System.out.print("Enter when");
        when=sc.nextLine();
        System.out.print("Enter rate");
        rate=sc.nextFloat();
        String checkBuyingStatus=buybook(buyer_id,book_id,when,rate);
        System.out.println(checkBuyingStatus);
    }
}

```

Use case: Sell Books

```

import java.util.*;
import java.sql.*;
public class Sell
{
    static Connection connection;
    static Statement statement;
    public Sell()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and load driver");
            System.exit(1);
        }
    }

    public static String sellbook(int s_id,int b_id,String day,int profit)
    {
        String check="Not Sold";
        try
        {
            connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","eesha","eeshasoham");

```

```

statement = connection.createStatement();
String query1="select s_id,b_id,day,profit from sells";
PreparedStatement ps=connection.prepareStatement(query1);
ResultSet rs=ps.executeQuery();
while(rs.next())
{
    int s=rs.getInt("s_id");
    int b=rs.getInt("b_id");
    String d=rs.getString("day");
    int p=rs.getInt("profit");
    if(s==s_id && b==b_id && d.trim().equals(day) && p==profit)
    {
        check="Already sold";
        break;
    }
}
if(check=="Already sold")
{
    check="Book already sold, choose another one";
}
else
{
    String query= "INSERT INTO sells
VALUES('"+s_id+"','"+b_id+"','"+day+"','"+profit+"')";
    int i = statement.executeUpdate(query);
    check="Successfully sold";
}
}
catch(Exception e)
{
    check="Exception or Invalid input.Please try again!";
    System.out.println(e);
}
return check;
}
public static void main(String[] args)
{
    Sell s=new Sell();
    int s_id,b_id;

```

```

String day;
int profit;
Scanner sc=new Scanner(System.in);
System.out.print("Enter s_id");
s_id=sc.nextInt();
System.out.print("Enter b_id");
b_id=sc.nextInt();
sc.nextLine();
System.out.print("Enter day");
day=sc.nextLine();
System.out.print("Enter profit");
profit=sc.nextInt();
String checkSellingStatus=sellbook(s_id,b_id,day,profit);
System.out.println(checkSellingStatus);
    }
}

```

CONFIGURATION MANAGEMENT

Description:

Configuration management helps in maintaining consistency of the project's functionalities with respect to its requirements. The screenshots below show three versions of code.

Every addition from the previous commit is shown in green colour and every deletion from the previous commit is shown in red colour. With every version the efficiency of the code is improved further, and the functionalities are served better.

To give a small example: The second version has an extra field called when (which refers to the date on which the book was bought) and also has a dialogue box which displays any errors if any may occur.

Use Case: Buy Books

1. Version 1

```
1=import java.awt.*;
2 import java.awt.event.*;
3 import java.sql.*;
4 public class BuyBooksV1 extends Frame
5 {
6     Button buyBB;
7     TextField buyIdTf, buyDtTf;
8     //TextArea errorText;
9     Connection connection;
10    Statement statement;
11=    public BuyBooksV1()
12    {
13        try
14        {
15            Class.forName("oracle.jdbc.driver.OracleDriver");
16        }
17        catch (Exception e)
18        {
19            System.err.println("Cannot find and load driver");
20            System.exit(1);
21        }
22        connectToDB();
23    }
24
25=    public void connectToDB()
26    {
27        ..
```

```

32 }
33 catch (SQLException connectException)
34 {
35     System.out.println(connectException.getMessage());
36     System.out.println(connectException.getSQLState());
37     System.out.println(connectException.getErrorCode());
38     System.exit(1);
39 }
40 }
41 public void buildGUI()
42 {
43     BuyBB = new Button("Buy Book");
44     BuyBB.addActionListener(new ActionListener()
45     {
46         public void actionPerformed(ActionEvent e)
47         {
48             try
49             {
50                 String query= "INSERT INTO buys VALUES(" + buidTf.getText() + ", " + boidTf.getText() + ")";
51                 int i = statement.executeUpdate(query);
52                 // errorText.append("\nInserted " + i + " rows successfully");
53             }
54             catch (SQLException insertException)
55             {
56                 // displaySQLErrors(insertException);
57             }
58         }
59     });
60 }
61
62 Panel first = new Panel();
63 first.setLayout(new GridLayout(4, 2));
64 first.add(new Label("Buyer ID:"));
65 first.add(buidTf);
66 first.add(new Label("Book ID:"));
67 first.add(boidTf);
68
69 first.setBounds(125,90,200,100);
70
71 Panel second = new Panel(new GridLayout(4, 1));
72 second.add(BuyBB);
73 second.setBounds(125,220,150,100);
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92 public static void main(String[] args)
93 {
94
95
96
97
98
99
100
101
102
103
104
105
106

```

GUI screenshot:



2. Version 2

<> Edit file

Preview changes

```

...   ...   @@ -4,8 +4,8 @@
4       4       public class BuyBooksV1 extends Frame
5       5       {
6       6       Button buyBB;
7       7       TextField buildTf, boldTf;
8       8       //TextArea errorText;
9       9       TextField buildTf, boldTf, whenTf;
10      10      TextArea errorText;
11      11      Connection connection;
12      12      Statement statement;
13      13      public BuyBooksV1()
14      14      {
15      15      {
16      16      try
17      17      {
18      18      String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ")";
19      19      String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ")"+whenTf.getText();
20      20      int i = statement.executeUpdate(query);
21      21      // errorText.append("\nInserted " + i + " rows successfully");
22      22      errorText.append("\nInserted " + i + " rows successfully");
23      23      }
24      24      catch (SQLException insertException)
25      25      {
26      26      // displaySQLErrors(insertException);
27      27      displaySQLErrors(insertException);
28      28      }
29      29      }
30      30      }
31      31      ...

```

@@ -47,30 +47,33 @@ public void actionPerformed(ActionEvent e)

```

47      47      {
48      48      try
49      49      {
50      50      String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ")";
51      51      String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ")"+whenTf.getText();
52      52      int i = statement.executeUpdate(query);
53      53      // errorText.append("\nInserted " + i + " rows successfully");
54      54      errorText.append("\nInserted " + i + " rows successfully");
55      55      }
56      56      catch (SQLException insertException)
57      57      {
58      58      // displaySQLErrors(insertException);
59      59      displaySQLErrors(insertException);
60      60      }
61      61      }
62      62      }
63      63      }
64      64      }
65      65      }
66      66      }
67      67      }
68      68      }
69      69      }
70      70      }
71      71      }

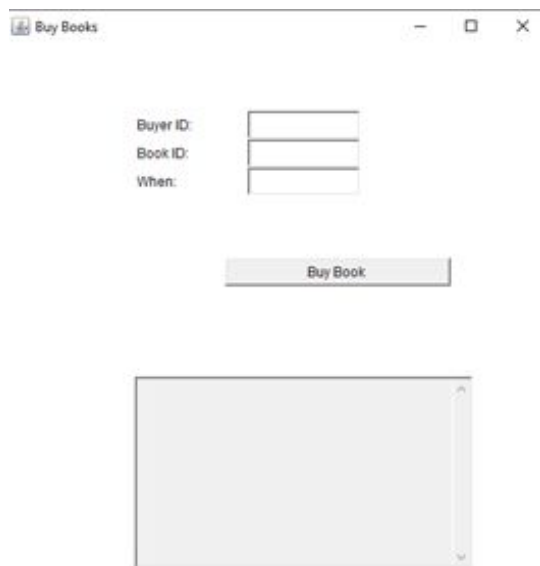
```

```

66 +         errorText = new TextArea(10, 40);
67 +         errorText.setEditable(false);
68
69 +         Panel first = new Panel();
70 +         first.setLayout(new GridLayout(4, 2));
71 +         first.add(new Label("Buyer ID:"));
72 +         first.add(new JTextField());
73 +         first.add(new Label("Book ID:"));
74 +         first.add(new JTextField());
75 +         first.add(new Label("When:"));
76 +         first.add(new JTextField());
77
78 +         first.setBounds(125, 90, 200, 100);
79
80 + @ -83,6 +91,15 @@ public void actionPerformed(ActionEvent e)
81 +     {
82 +         setVisible(true);
83 +     }
84
85 + private void displaySQLExceptions(SQLException e)
86 +     {
87 +         errorText.append("SQLException: " + e.getMessage() + "\n");
88 +         errorText.append("SQLState: " + e.getSQLState() + "\n");
89 +         errorText.append("VendorError: " + e.getErrorCode() + "\n");
90 +     }
91
92 + public static void main(String[] args)
93 +     {

```

GUI screenshot:



3. Version 3

<> Edit file		Preview changes	
...	...	@@ -4,7 +4,7 @@	
4	4	public class BuyBooksV1 extends Frame	
5	5	{	
6	6	Button buyBB;	
7	-	TextField buildTf, boldTf,whenTf;	
7	+	TextField buildTf, boldTf,whenTf,rateTf;	
8	8	TextArea errorText;	
9	9	Connection connection;	
10	10	Statement statement;	
...	...	@@ -47,7 +47,7 @@ public void actionPerformed(ActionEvent e)	
47	47	{	
48	48	try	
49	49	{	
50	-	String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ""+whenTf	
50	+	String query= "INSERT INTO buys VALUES(" + buildTf.getText() + ", " + boldTf.getText() + ""+whenTf	
51	51	int i = statement.executeUpdate(query);	
52	52	errorText.append("\nInserted " + i + " rows successfully");	
53	53	}	
...	...	@@ -61,7 +61,7 @@ public void actionPerformed(ActionEvent e)	
61	61	buildTf = new TextField(15);	
62	62	boldTf = new TextField(15);	
63	63	whenTf = new TextField(15);	
64	-		
64	+	rateTf = new TextField(15);	
...	...		
62	62	boldTf = new TextField(15);	
63	63	whenTf = new TextField(15);	
64	-		
64	+	rateTf = new TextField(15);	
65	65		
66	66	errorText = new TextArea(10, 40);	
67	67	errorText.setEditable(false);	
...	...	@@ -74,6 +74,8 @@ public void actionPerformed(ActionEvent e)	
74	74	first.add(boldTf);	
75	75	first.add(new Label("When:"));	
76	76	first.add(whenTf);	
77	+	first.add(new Label("rate:"));	
78	+	first.add(rateTf);	
79	79		
78	80	first.setBounds(125,90,200,100);	
79	81		
...	...	@@ -87,8 +89,11 @@ public void actionPerformed(ActionEvent e)	
87	89	add(second);	
88	90		
89	91	setTitle("Buying Books");	
92	+	Color clr = new Color(100, 100, 140);	
93	+	setBackground(clr);	
94	94	setSize(500, 600);	
95	95	setVisible(true);	
96	+		
97	97	}	

Use case: Sell Books

1. Version 1

```
121 Sell.java
... @@ -0,0 +1,121 @@
1 + import java.awt.*;
2 + import java.awt.event.*;
3 + import java.sql.*;
4 + public class Sell extends Frame
5 + {
6 +     Button sellBookButton;
7 +     TextField s_idText, b_idText;
8 +     TextArea errorText;
9 +     Connection connection;
10 +     Statement statement;
11 +     public Sell()
12 +     {
13 +         try
14 +         {
15 +             Class.forName("oracle.jdbc.driver.OracleDriver");
16 +         }
17 +         catch (Exception e)
18 +         {
19 +             System.err.println("Unable to find and load driver");
20 +             System.exit(1);
21 +         }
22 +         connectToDB();
23 +     }
24 +
25 +     public void connectToDB()
26 +     {
27 +         try
28 +         {
29 +             connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","eesha","eeshasoham");
30 +             statement = connection.createStatement();
31 +         }
32 +         catch (SQLException connectException)
33 +         {
34 +             System.out.println(connectException.getMessage());
35 +             System.out.println(connectException.getSQLState());
36 +             System.out.println(connectException.getErrorCode());
37 +             System.exit(1);
38 +         }
39 +     }
40 +
41 +     public void buildGUI()
42 +     {
43 +         //Handle Insert Account Button
44 +         sellBookButton = new Button("Sell Book");
45 +         sellBookButton.addActionListener(new ActionListener()
46 +         {
47 +             public void actionPerformed(ActionEvent e)
48 +             {
49 +                 try
50 +                 {
51 +
52 +                     String query= "INSERT INTO sells VALUES(" + s_idText.getText() + ", " + "'" + b_idText.getText() +
53 +                     int i = statement.executeUpdate(query);
54 +                     errorText.append("\nInserted " + i + " rows successfully");
55 +                 }
56 +                 catch (SQLException e)
57 +                 {
58 +                     System.out.println(e.getMessage());
59 +                     System.out.println(e.getSQLState());
60 +                     System.out.println(e.getErrorCode());
61 +                     System.exit(1);
62 +                 }
63 +             }
64 +         });
65 +     }
66 + }
```

```

55 +         }
56 +         catch (SQLException insertException)
57 +         {
58 +             displaySQLErrors(insertException);
59 +         }
60 +     }
61 + });
62 +
63 +
64 +     s_idText = new TextField(15);
65 +     b_idText = new TextField(15);
66 +
67 +
68 +     errorText = new TextArea(10, 40);
69 +     errorText.setEditable(false);
70 +
71 +     Panel first = new Panel();
72 +     first.setLayout(new GridLayout(4, 2));
73 +     first.add(new Label("Seller ID:"));
74 +     first.add(s_idText);
75 +     first.add(new Label("Book ID:"));
76 +     first.add(b_idText);
77 +
78 +     first.setBounds(125,90,200,100);
79 +
80 +     Panel second = new Panel(new GridLayout(4, 1));
81 +     second.add(sellBookButton);
82 +     second.setBounds(125,220,150,100);

```

```

93 +     setSize(500, 600);
94 +     setVisible(true);
95 + }
96 +
97 +
98 +
99 + private void displaySQLErrors(SQLException e)
100 + {
101 +     errorText.append("\nSQLException: " + e.getMessage() + "\n");
102 +     errorText.append("SQLState:      " + e.getSQLState() + "\n");
103 +     errorText.append("VendorError:  " + e.getErrorCode() + "\n");
104 + }
105 +
106 +
107 +
108 + public static void main(String[] args)
109 + {
110 +     Sell s = new Sell();
111 +
112 +     s.addWindowListener(new WindowAdapter(){
113 +         public void windowClosing(WindowEvent e)
114 +         {
115 +             System.exit(0);
116 +         }
117 +     });
118 +
119 +     s.buildGUI();
120 + }

```

GUI screenshot:



Seller ID:

Book ID:

Sell Book

2. Version 2

Showing 1 changed file with 11 additions and 4 deletions.

Unified Split

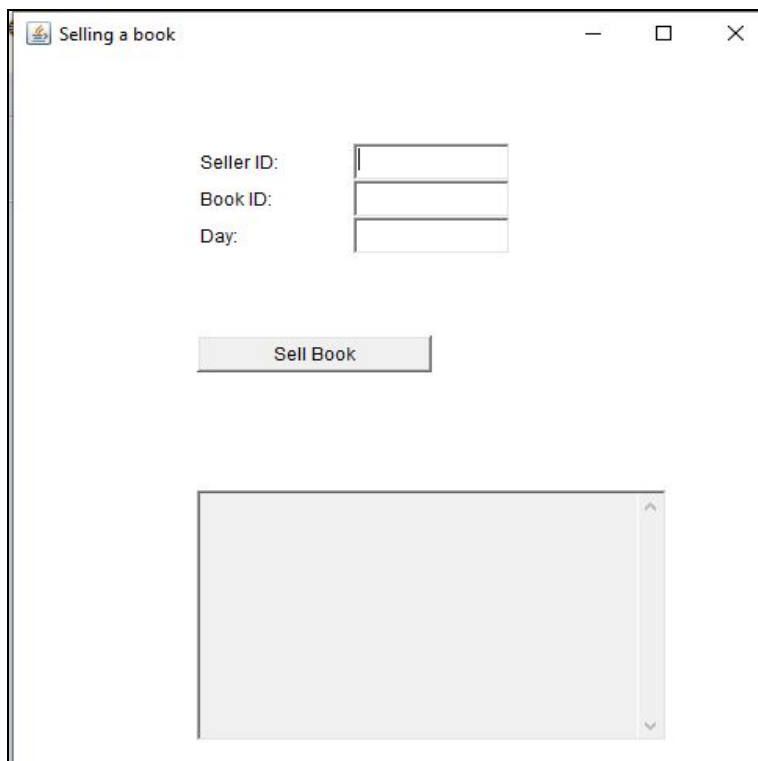
```
15 Sell.java
@@ -4,7 +4,7 @@
4 4 public class Sell extends Frame
5 5 {
6 6     Button sellBookButton;
7 7 - TextField s_idText, b_idText;
7 7 + TextField s_idText, b_idText, dayText;
8 8     TextArea errorText;
9 9     Connection connection;
10 10    Statement statement;
@@ -49,7 +49,7 @@ public void actionPerformed(ActionEvent e)
49 49        try
50 50        {
51 51
52 52 - String query= "INSERT INTO sells VALUES(" + s_idText.getText() + ", " + "" + b_idText.getText() +
52 52 + String query= "INSERT INTO sells VALUES(" + s_idText.getText() + ", " + "" + b_idText.getText() +
53 53        int i = statement.executeUpdate(query);
54 54        errorText.append("\nInserted " + i + " rows successfully");
55 55    }
@@ -63,6 +63,7 @@ public void actionPerformed(ActionEvent e)
63 63
64 64        s_idText = new TextField(15);
65 65        b_idText = new TextField(15);
66 66 +        dayText = new TextField(15);
```

```

68 69          errorText = new TextArea(10, 40);
@@ -74,22 +75,28 @@ public void actionPerformed(ActionEvent e)
74 75          first.add(s_idText);
75 76          first.add(new Label("Book ID:"));
76 77          first.add(b_idText);
78 +          first.add(new Label("Day:"));
79 +          first.add(dayText);
77 80
78 81          first.setBounds(125,90,200,100);
79 82
80 83          Panel second = new Panel(new GridLayout(4, 1));
81 84          second.add(sellBookButton);
82 -          second.setBounds(125,220,150,100);
85 +          second.setBounds(125,220,150,100);
86 +          Panel third = new Panel();
87 +          third.add(errorText);
88 +          third.setBounds(125,320,300,200);
83 89
84 90
85 91
86 92          setLayout(null);
87 93
88 94          add(first);
89 95          add(second);
96 +          add(third);
90 97
91 98
92 -
99 +          setTitle("Selling a book");
93 100          setSize(500, 600);
94 101          setVisible(true);

```

GUI screenshot:



3. Version 3

Showing 1 changed file with 11 additions and 3 deletions.

Unified Split

```
14 Sell.java

@@ -4,7 +4,7 @@
4 4 public class Sell extends Frame
5 5 {
6 6     Button sellBookButton;
7 - TextField s_idText, b_idText, dayText;
7 + TextField s_idText, b_idText, dayText, profitText;
8 8     TextArea errorText;
9 9     Connection connection;
10 10    Statement statement;

@@ -49,7 +49,7 @@ public void actionPerformed(ActionEvent e)
49 49         try
50 50         {
51 51
52 - String query= "INSERT INTO sells VALUES(" + s_idText.getText() + ", " + "" + b_idText.getText() + ", " + "" + dayText.getText() + ")";
52 + String query= "INSERT INTO sells VALUES(" + s_idText.getText() + ", " + "" + b_idText.getText() + ", " + "" + dayText.getText() + ", " + "" + profitText.getText() + ")";
53 53         int i = statement.executeUpdate(query);
54 54         errorText.append("\nInserted " + i + " rows successfully");
55 55     }

@@ -63,7 +63,9 @@ public void actionPerformed(ActionEvent e)
63 63
64 64     s_idText = new TextField(15);
65 65     b_idText = new TextField(15);
66 - dayText = new TextField(15);
66 + dayText = new TextField(15);
67 + profitText = new TextField(15);
68 +

67 69
68 70
69 71     errorText = new TextArea(10, 40);

@@ -77,6 +79,9 @@ public void actionPerformed(ActionEvent e)
77 79     first.add(b_idText);
78 80     first.add(new Label("Day:"));
79 81     first.add(dayText);
82 + first.add(new Label("Profit:"));
83 + first.add(profitText);
84 +

80 85
81 86     first.setBounds(125,90,200,100);
82 87

@@ -97,6 +102,9 @@ public void actionPerformed(ActionEvent e)
97 102
98 103
99 104     setTitle("Selling a book");
105 + Color clr = new Color(230, 190, 250);
106 + setBackground(clr);
107 + setFont(new Font("Cambria", Font.BOLD, 15));
108 108     setSize(500, 600);
109 109     setVisible(true);
110 110 }
```

GUI screenshot:

The screenshot shows a window titled "Selling a book" with a light purple background. It contains four input fields for "Seller ID:", "Book ID:", "Day:", and "Profit:". Below these fields is a button labeled "Sell Book". At the bottom of the window is a large, empty rectangular area, likely for a description or image.

Seller ID:	<input type="text"/>
Book ID:	<input type="text"/>
Day:	<input type="text"/>
Profit:	<input type="text"/>

TESTING

Test Plan

List of the test cases proposed:

1. Book availability
2. Request for notification when the book becomes available
3. Check if a particular book can be sold
4. Check the payment status
5. Donate money to the required charity/organization/person
6. Check for availability of donation
7. Upload the information/article
8. Verify the information

Test cases

Description:

Given below are the sample test cases for the four major use cases in our project. The four use cases are namely, buy books, sell books, raise funds and raise awareness. Our project is essentially an online kindle and a platform that serves as a place for growth and development.

Each of the four use cases have two test cases each and the details about each test case is discussed in brief below. The pre and post conditions of these test cases are also discussed below.

Test cases under each use case:

- For buy books use case
 - Book availability
 - Request for notification when the book becomes available
- For sell books use case
 - Check if a particular book can be sold
 - Check the payment status

- For raise funds use case
 - Donate money to the required charity/organization/person
 - Check for availability of donation
- For raise awareness use case
 - Upload the information/article
 - Verify the information

1. Buy Books Use Case:

#Test case 1

Test Case ID: 1.1

Test case name: To test if a certain book is available for the user to buy

Subsystem: Buy Books

Tester's Name: Gayatri

Software Version: 1.0

Operating System: Windows

Date of Test: 02-05-2020

Initial setup:

The idea of this test case is to check the availability of the book the user wants to buy. So, the initial step would be, for the user to search for a book of their choice and then enter its details to check for availability.

Input:

Details of the book such as the book's name, rate, author and so on will be entered by the user.

Expected results:

The result is expected to be, book “available” if the book is indeed available for buying. Alternatively, if the book is not accessible by the user due to any reason, the output is expected to be “Unavailable”.

Actual Results:

The test case was successfully executed.

#Test case 2

Test Case ID: 1.2

Test Case name: Request for notification when book is available

Subsystem: Buy Books

Tester’s Name: Gayatri

Software Version: 1.0

Operating System: Windows

Date of Test: 01-05-2020

Initial setup:

The initial step in this test case will be the user searching for books they might be interested in. Then, they will have to enter the details into the search bar and check for availability. If the book is unavailable to be bought/borrowed, the user can proceed to avail the “notify when available” option.

Input:

The user will have to check for availability by entering the details and then if the book is unavailable then they will have to tick the button that says “notify when available”.

Expected results:

The expected result of this test case is for a message that states, ”Grow Together will notify when the book becomes available” should be displayed.

Actual results:

This test case passed the review and the expected result was obtained.

2. Sell Books Use Case:

#Test case 1

Test Case ID: 2.1

Test Case Name: Check the possibility of the book to be sold.

Subsystem: Sell books

Tester's Name: Eesha

Software Version: 1

Operating System: Windows

Date of Test: 02 May

Initial Setup:

So, basically the idea of this test case is to check the possibility of the book to be sold. The seller should first get himself verified as a member and then enter the details of the book and put it up for sale. Only if the book is visible to the buyer, he/she can place an order for it.

Input:

The next step would be entering the book details and checking if there are any orders for that particular book.

Expected Results:

When the seller checks up on any book, he should get the count of the orders with date and time.

Actual Results:

Pass

#Test case 2

Test Case ID: 2.2

Test Case Name: Check payment status

Subsystem: Sell books

Tester's Name: Eesha

Software Version: 1

Operating System: Windows

Date of Test: 02 May

Initial Setup:

So, basically the idea of this test case is to check the payment status. The seller should first get himself verified as a member and then browse the 'orders' section and check if he has any pending orders or not.

Input:

The next step would be selecting an order from the list of pending orders and checking if the payment for the book has been completed or not.

Expected Results:

When the seller selects a particular order he should get the payment status of the order. If the payment has been completed, a pop up message saying "Payment received, proceed to sell" should be displayed otherwise "Payment is pending" should be displayed.

Actual Results:

Pass

3. Raise Funds Use Case

#Test case 1

Test Case ID: 3.1

Test Case Name: To donate money to the required charity

Subsystem: Raise Funds

Tester's Name: Gayatri

Software version: 1.0

Operating system: Windows

Date of test: 03-05-2020

Initial setup:

The idea of this test case is to let the user to donate funds to charities, emergencies and any other such situations. So, the initial step would require the user to chose the charity/organization/person they want to donate to.

Input:

The input required for donating would be the bank/card details or in simpler words, filling out the payment method information.

Expected result:

The test case is to check if the user is successfully able to donate to the charity/organization/person they want to. So, the expected output would be either “Donated Successfully” or “Donation Unsuccessful” in case of a failure.

Actual output:

The review of this test case was successful, and the expected result was indeed obtained.

#Test case 2

Test case ID: 3.2

Test case name: Check for availability of donation

Subsystem: Raise Funds

Tester's name: Gayatri

Software Version: 1.0

Operating System: Windows

Date of Test: 03-05-2020

Initial setup:

The idea of this test case is to notify the user if the charity/organization/person they want to donate to is available. So, the initial setup will require for the user to search for charities/people they want to donate to.

Input:

The input here is for the user to press the button for checking the availability of the donation.

Expected results:

The expected result of this test case is for the message “This charity/organization/person is accepting donations now” in case it is and “This charity/organization/person is not accepting donations anymore” in case it is not.

Actual output:

The review of this test case was successful, and the expected result was indeed obtained.

4. Raise Awareness Use Case:

#Test case 1

Test Case ID: 4.1

Test Case Name: Upload the article/information

Subsystem: Raise Awareness

Tester's Name: Eesha

Software Version: 1

Operating System: Windows

Date Of Test: 04 May

Initial Setup:

So, basically the idea of this test case is to check if a person is able to upload an article successfully or not. Initially, the person should be a member to be able to post anything. The user must login and get himself/ herself verified and then he/she can proceed further.

Input:

The next step would be uploading the document for spreading awareness among people who have little or no knowledge about what is happening in and around the country.

Expected Results:

When the person finishes uploading the document, he/she should get a pop up message saying “Article uploaded successfully”.

Actual Results:

Pass

#Test case 2

Test Case ID: 4.2

Test Case Name: Verify the information

Subsystem: Raise Awareness

Tester's Name: Eesha

Software Version: 1

Operating System: Windows

Date of Test: 04 May

Initial Setup:

So, basically the idea of this test case is to check if the information given by a particular member is genuine or not because this is not the place for fake news. For this to happen, the member has to upload an article in the first place.

Input:

The next step would be verifying the information of the article from a number of sources and validating the article.

Expected Results:

When the validation is complete, a pop up box should be displayed saying “Make it public”. If the validation fails, then a message saying “Cannot be uploaded” should be displayed.

Actual Results:

Pass

JUnit Testing outputs

Description:

Use case being tested: Buy Books

Tester’s name: Gayatri

About the testing:

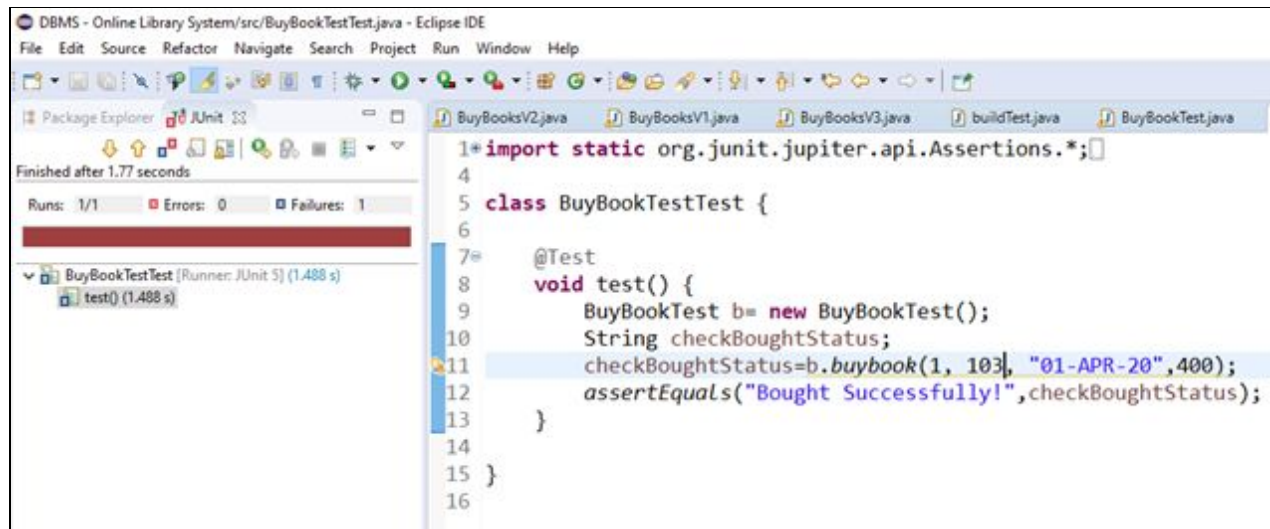
For the use case buying of books, we are creating a simple j unit test case to check the working of this code. Here the code is being checked by entering the details of the buyer and book (book details include: when it is being bought and the price of the book), this data is being cross checked with the data in the database to see if the user can buy this book.

If the book is available for being bought the test case returns success and If it isn’t it returns an error.

Use case: Buy

Output Screenshots:

Negative result for Junit test:



```
1*import static org.junit.jupiter.api.Assertions.*;
4
5 class BuyBookTestTest {
6
7     @Test
8     void test() {
9         BuyBookTest b= new BuyBookTest();
10        String checkBoughtStatus;
11        checkBoughtStatus=b.buybook(1, 103, "01-APR-20",400);
12        assertEquals("Bought Successfully!",checkBoughtStatus);
13    }
14
15 }
16
```

DBMS - Online Library System/src/BuyBookTestTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit

Finished after 1.77 seconds

Runs: 1/1 Errors: 0 Failures: 1

BuyBookTestTest [Runner: JUnit 5] (1.488 s)

test() (1.488 s)

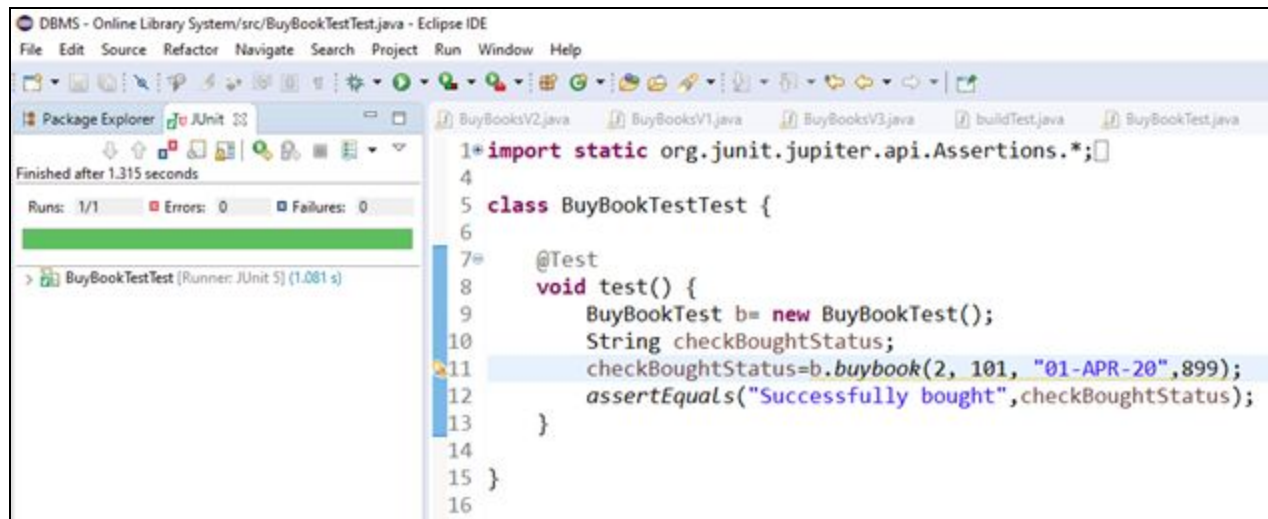
Here the user 1, is trying to buy book with ID 103 and the Junit returned error, clearly indicating the user cannot buy this particular book.

This error occurred as the book was already previously bought by the user. Since the data was already stored in the database, the request to buy the book again was denied.

BUYER_ID	BOOK_ID	WHEN	RATE
1	103	01-APR-20	400

Screenshot showing the data that the buyer has previously already purchased the book with ID 103.

Positive result for Junit test:



The screenshot shows the Eclipse IDE with a JUnit test named `BuyBookTestTest` passing. The test code is as follows:

```
1 *import static org.junit.jupiter.api.Assertions.*;
4
5 class BuyBookTestTest {
6
7     @Test
8     void test() {
9         BuyBookTest b= new BuyBookTest();
10        String checkBoughtStatus;
11        checkBoughtStatus=b.buybook(2, 101, "01-APR-20",899);
12        assertEquals("Successfully bought",checkBoughtStatus);
13    }
14
15 }
16
```

The test results pane on the left shows "Finished after 1.315 seconds" and "Runs: 1/1", "Errors: 0", "Failures: 0".

Here the user 2 is trying to buy the book with ID 101 and is clearly successful in buying the book. Hence, the Junit test returned success.

The user with ID 2 had previously not purchased the book with ID 101 and thus was able to successfully make this purchase.

BUYER_ID	BOOK_ID	WHEN	RATE
1	103	01-APR-20	400
1	106	04-APR-20	646.34
2	102	08-MAR-20	650
3	104	19-APR-20	700
3	105	03-APR-20	498.45
4	108	31-MAR-20	859
5	107	18-MAR-20	540
5	101	02-APR-20	899
1	101	01-APR-20	899
2	104	23-APR-20	700
1	104	01-APR-20	400

The values before the Junit test was performed and clearly the User with ID 2 had not purchased book with ID 101.

BUYER_ID	BOOK_ID	WHEN	RATE
2	101	01-APR-20	899

The newly added value after the Junit test was performed.

The above screenshot is indicating that the user with ID 2 had not priorly purchased this book and thus since the same exact data was not present in the database the purchase was successful.

Description:

Use case being tested: Sell Books

Tester's name: Eesha

About the testing:

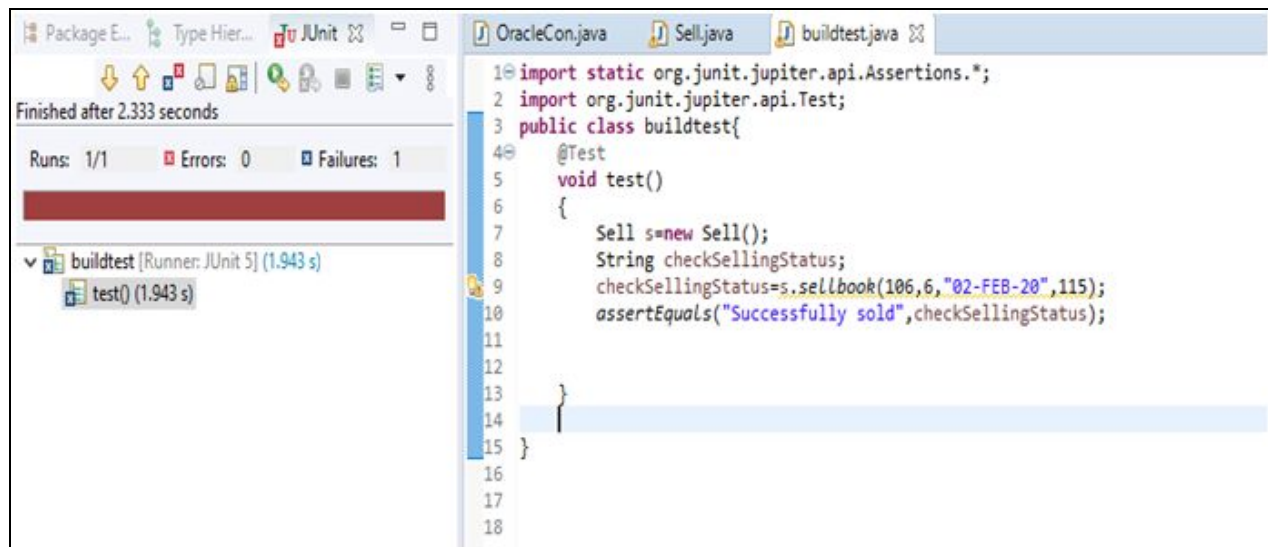
For the use case selling of books, we are creating a simple j unit test case to check the working of this code. Here the code is being checked by entering the details of the seller and book like the id, date and profit, this data is being cross checked with the data in the database to see if the user can sell this book.

If the book is available for being bought the test case returns success and If it isn't it returns an error.

Use case: Sell

Output Screenshots:

Negative result for Junit test:



```
1 import static org.junit.jupiter.api.Assertions.*;
2 import org.junit.jupiter.api.Test;
3 public class buildtest{
4     @Test
5     void test()
6     {
7         Sell s=new Sell();
8         String checkSellingStatus;
9         checkSellingStatus=s.sellbook(106,6,"02-FEB-20",115);
10        assertEquals("Successfully sold",checkSellingStatus);
11
12
13    }
14
15 }
```

Finished after 2.333 seconds

Runs: 1/1 Errors: 0 Failures: 1

buildtest [Runner: JUnit 5] (1.943 s)

test() (1.943 s)

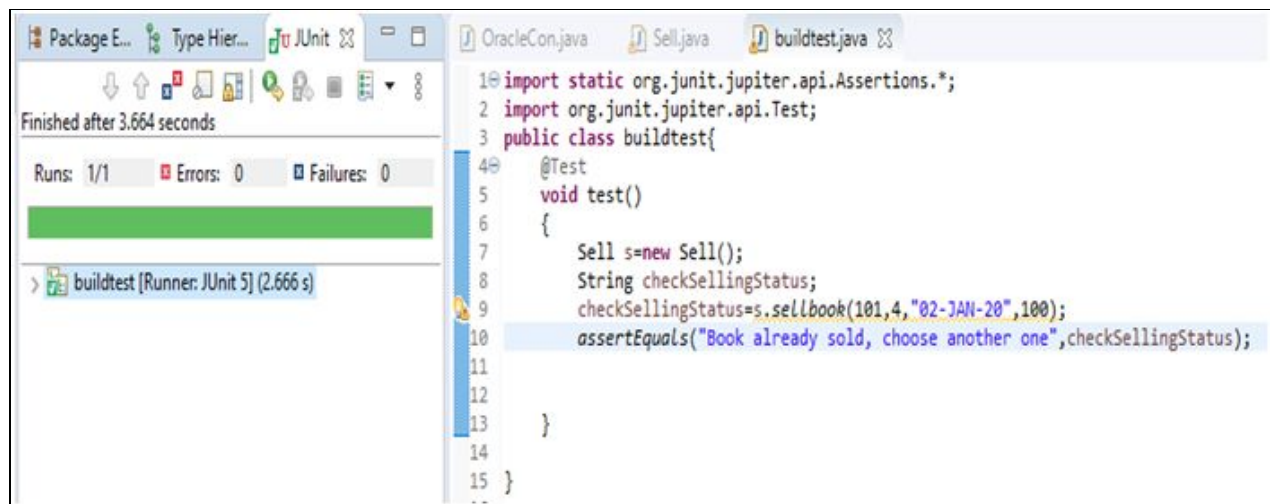
Here the seller tries to sell a particular book and the test case fails indicating that the book cannot be sold.

The problem is that the seller is trying to sell a book that is either not in stock or is not yet added to the display section.

B_ID	B_NAME	AUTHOR
1	harry potter	jk rowling
2	pride and prejudice	jane austin
3	the kite runner	khaled hosseini
4	book thief	markus zusak
5	alchemist	paulo coelho

A book with id=6 doesn't exist.

Positive result for Junit test:



Here the test case passes indicating that this feature is working perfectly fine.

The book that the seller is trying to sell is already sold and this is known with the help of the book id which is unique for every book. Hence, the seller receives a message that the bold has already been sold.

S_ID	B_ID	DAY	PROFIT
102	3	19-DEC-19	50
103	5	20-FEB-20	150
104	2	11-MAR-20	210
105	1	14-MAR-20	90
101	4	02-JAN-20	100

Book with id=4 is already sold by the seller with id=101.

CONTENT MANAGEMENT

Description:

The content management done using Drupal is a pictorial representation of how we want our system to look like. There is nothing over the top about it, we just wanted it to be as user friendly as possible.

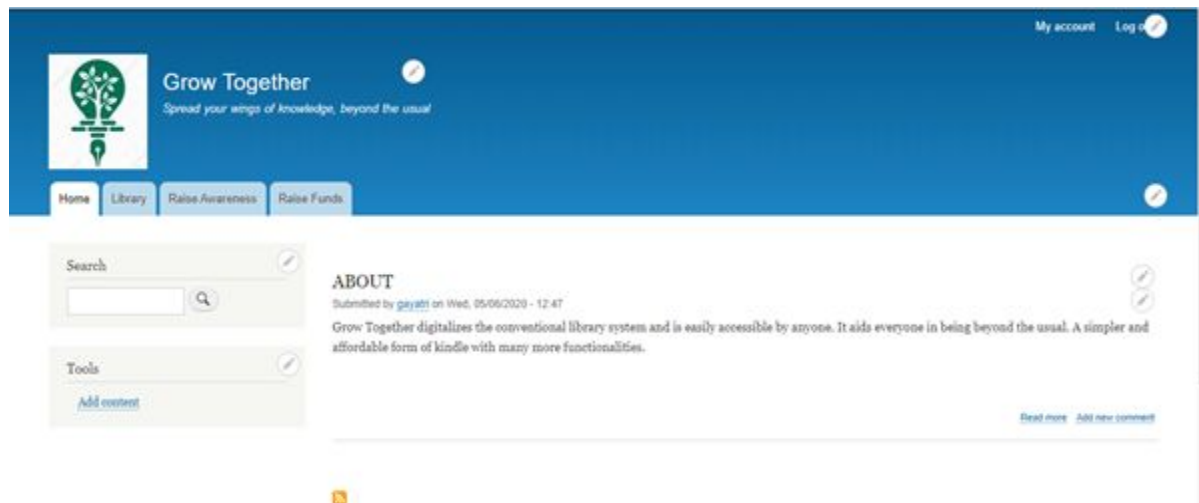
So, when the user logs in, he/she will be directed to the home page. Here in the 'about' section the user will get a brief idea as to how Grow Together works, what it is all about and what it has to offer.

Other than this there are various tabs which make it easier for the user to browse through. A search bar is also available which aids the user in searching for any specific information he/she may want.

There are functionally specific tabs so that the users can have a clear picture. For example, the raise funds tab will have the information of the various cases for which the funds are being collected, it will show the daily fund collections for each case, the target amount, the details of the organization/person who started the charity and so on.

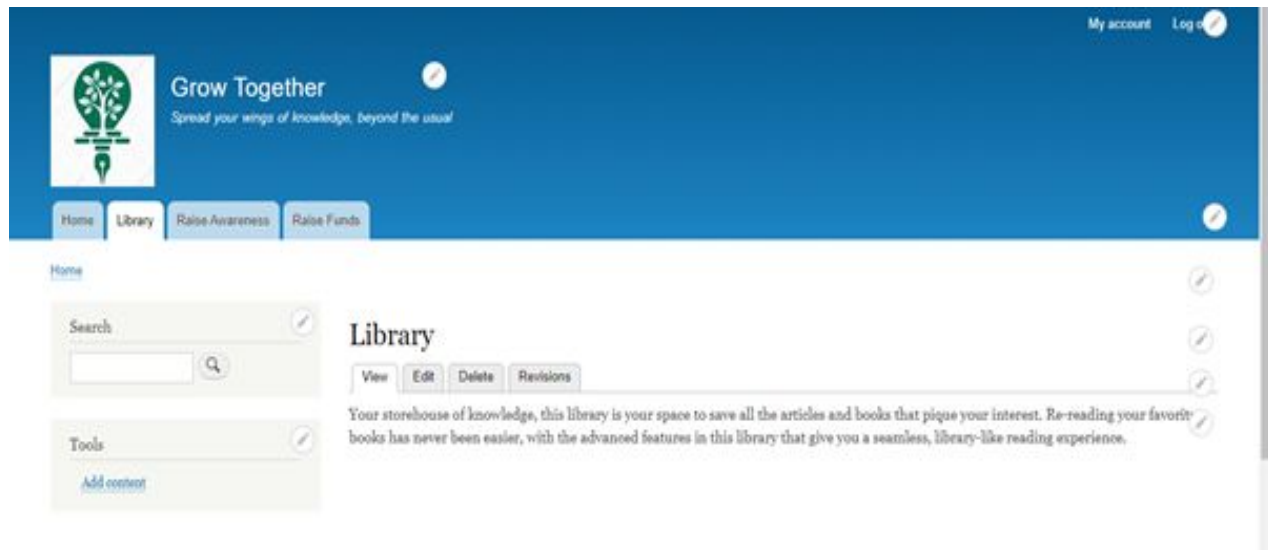
An additional feature is the library tab which stores the collection of all the books that you have read and purchased/rented.

A picture of how we want our system to look like:

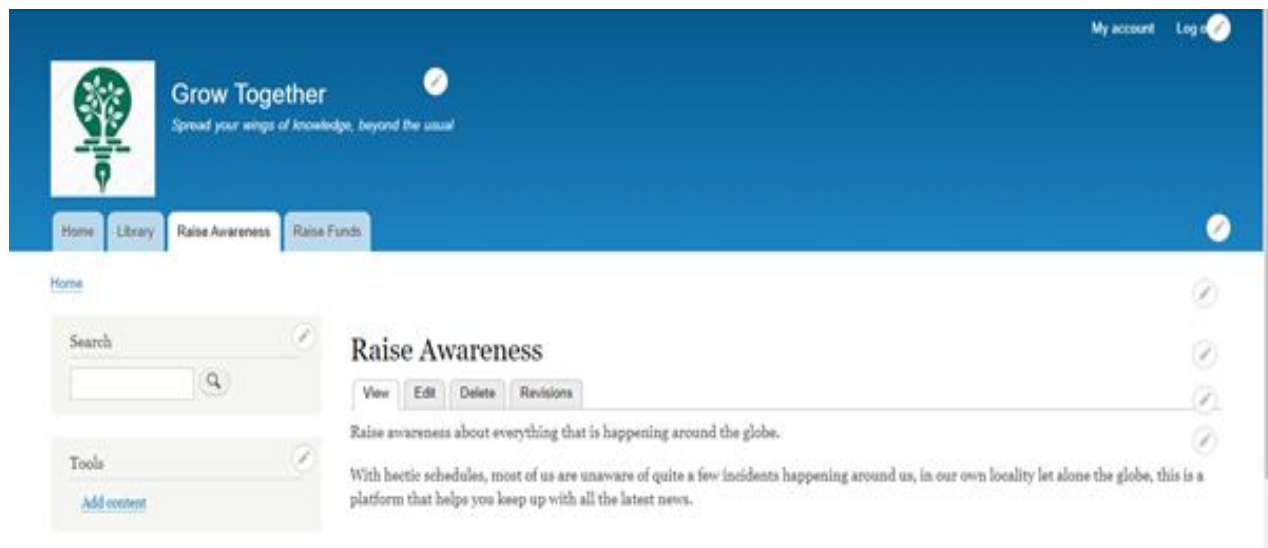


This is the home page containing the title, logo, a brief description and all the other tabs that the user can explore.

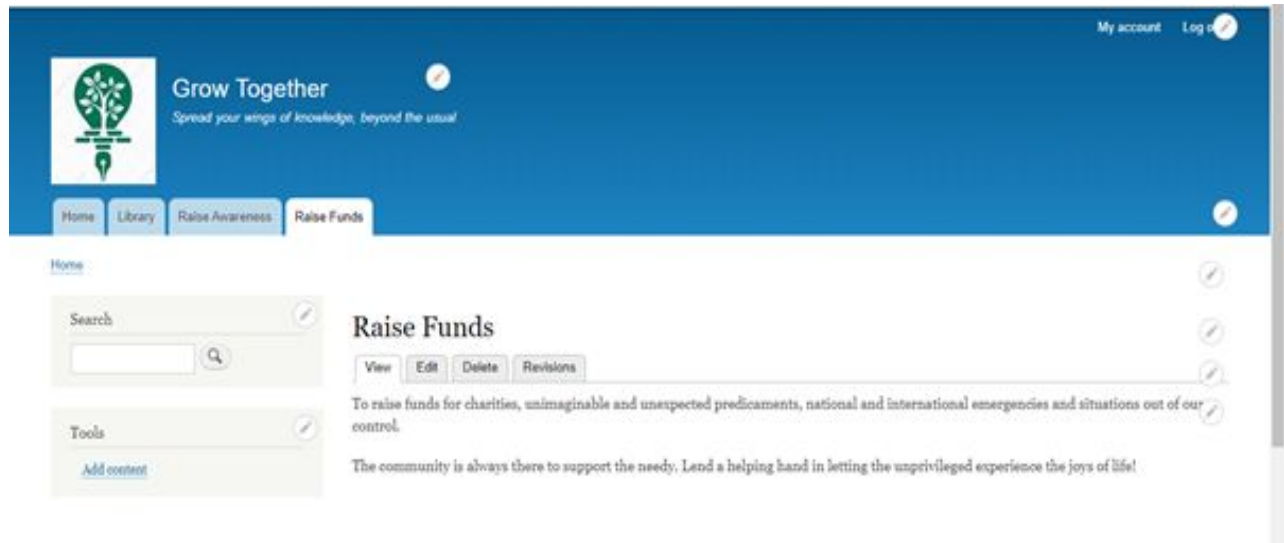
Other tabs:



The above screenshot show the library tab and a brief description about it.



The above screenshot shows the raise awareness tab and its description.

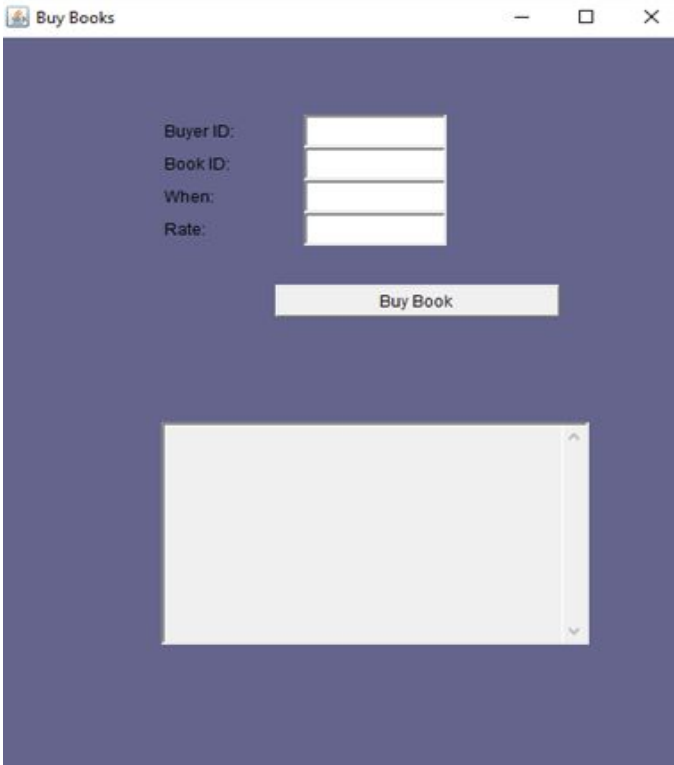


The above screenshot shows the raise funds tab and its description.

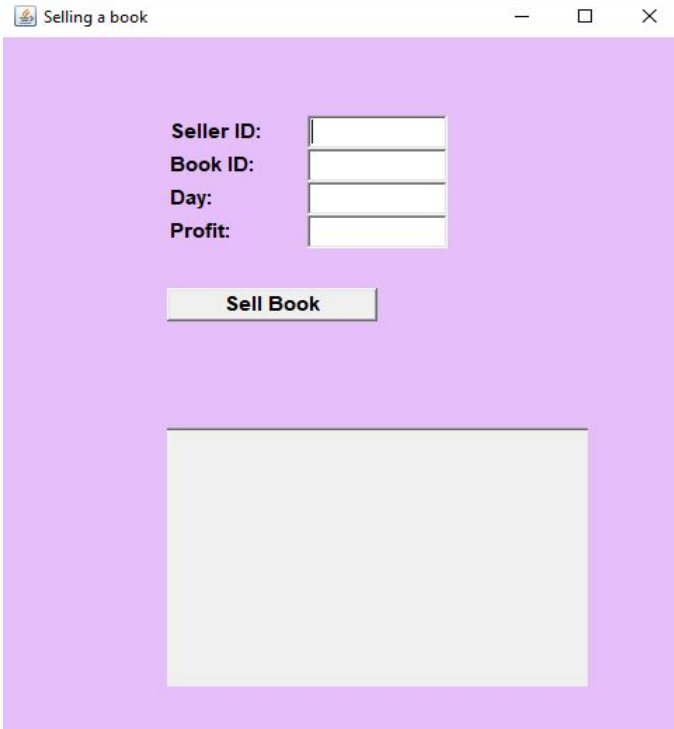
This is the simple and basic system we created for our project.

OUTPUT

GUI Screenshots:

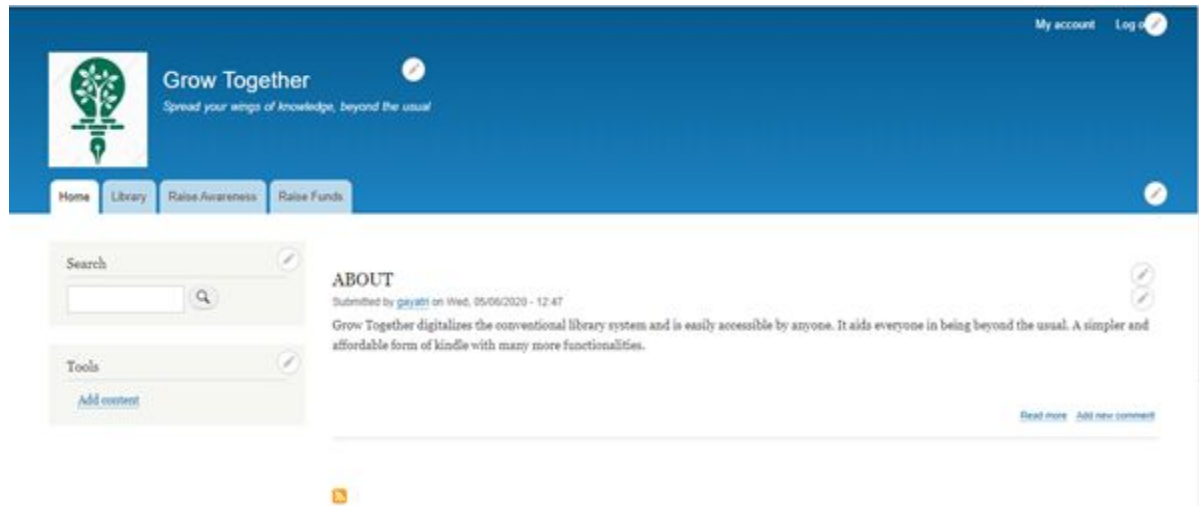


The 'Buy Books' window has a dark blue background. It contains four labels on the left: 'Buyer ID:', 'Book ID:', 'When:', and 'Rate:'. To the right of these labels are four white text input fields stacked vertically. Below the input fields is a light gray button labeled 'Buy Book'. At the bottom of the window is a large, empty white rectangular area with a vertical scrollbar on its right side.



The 'Selling a book' window has a light purple background. It contains four labels on the left: 'Seller ID:', 'Book ID:', 'Day:', and 'Profit:'. To the right of these labels are four white text input fields stacked vertically. Below the input fields is a light gray button labeled 'Sell Book'. At the bottom of the window is a large, empty white rectangular area with a vertical scrollbar on its right side.

A picture of how we want our system to look like:



CONCLUSION

This project, Grow Together, was created with the intention to develop a platform where a user could rent, buy and sell books. Additional functionalities were also included such as letting users raise funds and awareness about certain topics of their interest and of relevance. The project was initially developed using basic Java programming and SQL. The GUI thus obtained is shown in previous parts of this document. This project was modelled using the different data models available. Testing was also performed on the system and the outputs obtained are mentioned. Drupal was used as the content management system to obtain a rough idea about how the proposed system may look. After a thorough development process, the requirements and the basic notion behind Grow Together has been implemented successfully.

REFERENCES

- www.google.com
- www.github.com
- www.wikipedia.com
- www.youtube.com
- www.ieee.org
- www.bitnami.com