



A Website for Software Component Reuse Repository

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Computer Science & Engineering

by
**Ravinder Sokhal, Tanish Yadav,
Robin Singh Barala and Jyotirmoy Barman**
Group: CS-51

to the
**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD
June, 2020**

UNDERTAKING

I declare that the work presented in this report titled “A *Website for Software Component Reuse Repository*”, submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, for the award of the ***Bachelor of Technology*** degree in ***Computer Science & Engineering***, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

June, 2020

Allahabad

(Ravinder Sokhal)

(Tanish Yadav)

(Robin Singh Barala)

(Jyotirmoy Barman)

CERTIFICATE

Certified that the work contained in the report titled “*A Website for Software Component Reuse Repository*”, by *Ravinder Sokhal, Tanish Yadav, Robin Singh Barala and Jyotirmoy Barman*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Prof. Dharmendra Kumar Yadav)
Computer Science and Engineering Dept.
M.N.N.I.T. Allahabad

June, 2020

Preface

Software Development has greatly flourished in our generation. It has been forty years since the idea of developing software components was envisioned. Component-Based Software Engineering (CBSE) has attracted a lot of attention amongst the researchers and has led to many research results being published in the research literature. The three important factors of Software Development are costs, time-to-market, and quality products which affect the software industry. So the main objective of Component-Based Software Development (CSBD) is to write once and reuse the component any number of times with no or minor modification.

Based on the literature study we proposed a complete model for reusable Component-Based Software Development. This Model will cover both component-based software development as well as Component development phases.

Acknowledgments

We would like to thank everyone who contributed for the completion of our project. We are especially grateful to our mentor Prof. Dharmendra Kumar Yadav for assisting us in our project. Without their expertise and suggestions we would never have been able to complete our work.

We would also like to thank DUGC, Computer Science and Engineering Department, Dr. Shashank Shrivastava for his exemplary support and motivation for bringing about the possibility of projects for CSE/IT 2019-2020.

Abstract

Software Development marched its way from a traditional model-based approach to plug and play a component-based approach which enhanced the re-usability of the system for robust development. The main objective of Component-Based Software Development (CSBD) is to be able to reuse the components for the use of more complex systems with no or minor modification. Since each and every component is an independent executable unit that has been tested and deployed. Hence these components can be assembled together to form new software.

Contents

Preface	iv
Acknowledgments	v
Abstract	vi
1 Introduction	1
1.1 Motivation	2
1.2 Objective	3
1.3 Basic Concepts of CSBE	3
1.3.1 What are Software Components?	3
1.3.2 What Differentiates CBSE from the Conventional Reuse?	3
2 Related Work	5
3 Proposed Work	7
3.1 Overview of Our Work	7
3.2 Requirements	8
3.2.1 System Requirements	8
3.2.2 Functional Requirements	10
3.3 Specifications	12
3.3.1 User Interface	12
3.3.2 Preliminary Design	13
3.3.3 Object Interface	19
3.3.4 Formatting queries	22

3.4	Domain Analysis	23
4	Conclusion and Future Work	26
4.1	Conclusion	26
4.2	Future Work	26
	References	28

Chapter 1

Introduction

Software Industry has greatly flourished in the current few years and in the present Information Technology era, there is an enormous pressure in meeting the product deadlines with minimum development time and minimum development cost. And hence many software companies are adopting Component-based Development (CBD) to meet the demands of end-users for the products with changing requirement and at lower cost.

Component-based software engineering (CBSE) is used to develop/assemble software from existing components. Component-Based Software Engineering (CBSE) promotes the development of software systems through construction from existing software components, the development of components as reusable entities, and system evolution realization by the customization and replacement of components[8]. The idea of CBSE is not something new and has been envisioned more than forty years ago by McIlroy (NATO Conference, 1968). He provided an idea that the commercial component production similar to that found in other engineering fields.

Component-based software development (CSBD) helps in improving the maintainability, reliability and overall quality of software systems with reduced cost and time-to-market. The life cycle and software engineering model of CSBD is much different from that of the traditional ones. The inner structure of the component and the implementation of the interfaces are abstracted and provided with an interface for the proper functioning of the component. Therefore, CBSE, enables a distributed

and independent development of components as well as a straightforward replacement of a component by a different component in large-scale systems[5]. Based on the previous versions of Component-based software repositories we proposed a complete model for reusable component repository.

1.1 Motivation

The motivation for this work is based on advantages of component-based software engineering (CBSE) [6] and demands for such a reusable component repository.

In today's technological era there is enormous need to meet product deadline with minimum development time and minimum development cost. Reuse is the key paradigm for increasing software quality in software development. Using components to construct a product rather than building it from scratch is smarter and more efficient and profitable way. Having to spend minimum time in product development also reduces development cost.

To provide a picture of demand for such a reuse repository, here is a valid demand put up by a potential client in [stackoverflow.com](#) -

"I'm working as a software engineer for a company. We are going to apply some software engineering standards in our development process. We need a tool which provides a repository for our peripheral products (functions, classes, libraries, ...) which is created during the software development process for later use. The tool should provide some functionalities (e.g Name of the component, it's functionality, within which projects it is used?, author, publication date, list of known bugs, user rating, comment, ...) and it's better to have a web-based interface. Does anybody know such a software?"

[Link][4]

1.2 Objective

Objective of our work is to implement the component-based software engineering to make a web-based reusable component repository with several components which are implemented in many classical systems in our daily life. The main focus is to make these components easily available and provide the end-users with a functional interface and a default user interface as per their needs. We'll be implementing Component-based Software Development using Maven and Spring Framework on Tomcat server with a MySQL server as back-end.

1.3 Basic Concepts of CSBE

1.3.1 What are Software Components?

There can be several definitions of Software Components, however, in the context of CBSE, we emphasis **plug & play** software components so that software can be composed of several other components independent of production, acquisition and deployment that interact to form a functioning system [7].

Each software component can be comprised of one or more other smaller software components. These software components should define the semantics of the component, syntax and composition of the components as for better understanding for the end-users.

1.3.2 What Differentiates CBSE from the Conventional Reuse?

- Conventional Software Reuse

In the last several years object-oriented technologies promoted software reuse but there were still several issues with the whole systems

and class. These software were created according to the needs of the end-users and hence which made them specific and generalization wasn't done. Also in cases like this, the components were hard to compose together due to processing and other factors.

- CBSE Approach

The approach of CBSE is quite different than that of the Conventional Software Reuse.

- *Plug & Play*: The components should be portable, flexible and easy to understand as per the end-user's perspective. These components must be easy to handle such that they can be composed at run-time without compilation.
- *Interface-centric*: Proper abstraction and implementation of interface should be provided with the components such as to hide the complexity of the code and implementation details so that they can be composed without knowing their implementation.
- *Architecture-centric*: These components must be designed in a pre-defined architecture such that the components composed would easily inter-operate with each other. And these components working should not clash with any other components.
- *Standardization*: The interface provided for the component should be standardized so that they can be reused or manufactured by other vendors or corporations.
- *Distribution through Market*: Components can be acquired and improved with the competition in market and incentives must be provided to the vendors.

Chapter 2

Related Work

The reuse of software has been practiced ever since programming began. However, the reuse of software components is often traced to Doug Mcilroy's proposed paper based on the reusable components in software industry. Active areas of reuse research includes reuse of libraries, tools, designs, design patterns, business, finance, etc.

The ultimate purpose of domain engineering and systematic software reuse is to improve quality of the products and services that a company provides and, thereby, maximize profits. The Development of Components are focused on building reusable units which can be further integrated to form a specific system using several components.

Many component models have been developed like JavaBeans, EJB, COM, CCM, Koala, SOFA, KobrA, ADLs, UML2.0, PECOS, Pin and Fractal. In CBSE, composition is a central issue, since components are supposed to be used as building blocks from a repository and assembled or plugged together into larger blocks or systems[11].

Some of the related work are:

1. *Orca: A component model and Repository* [9]

Orca project was distinguished by explicit adoption of component-based approach from start and also acknowledged the relevance of

component market. In terms of development tool Orca traded-off portability for ease of use. Orca presented a model and Repository for building robotic wares.

2. *Enhanced Software Development for Reuse Process Model in Component Based Software Engineering* [10]

The goal was to refine reuse model that can promote CBSE. The researcher developed enhanced software development process for reuse and gave the future prospect of CBSE

Chapter 3

Proposed Work

3.1 Overview of Our Work

Our work is related to Component-based software development, for our work we're developing a repository for reusable components which can be used as building blocks for a better and specific system. We have used Maven, Spring Framework, Tomcat Server, MySQL server and JSP pages to build the interface for our website and all the models and components have been tested and deployed accordingly. Due to lack of time, we couldn't undergo alpha and beta testing for our models which we consider doing in our future work. We've created our components in such a way that they can be modified according to the choice of the end-user. Considering that there are components which is comprised of several other components and hence the end-user can also choose among those components. This website not only provides the end-users with a functional interface but also with an user interface as JSP pages which is very much easily designable according to the style sheet(CSS) they use. All the settings for the data source, i.e. the data from the database server are stored in beans inside an XML file. Since the configurations files are in XML format, therefore they can be easily understandable to the end-users who might not even have knowledge about the technologies

or complexity used in these components. All the components are developed in an object-oriented approach and with an functional interface which allows the end-users to use the following methods available for the objects, making it easier for the end-user without having to deal with the complexity of the component. The figure below explains how the component-based software engineering works and how the components act as building blocks for the required system.

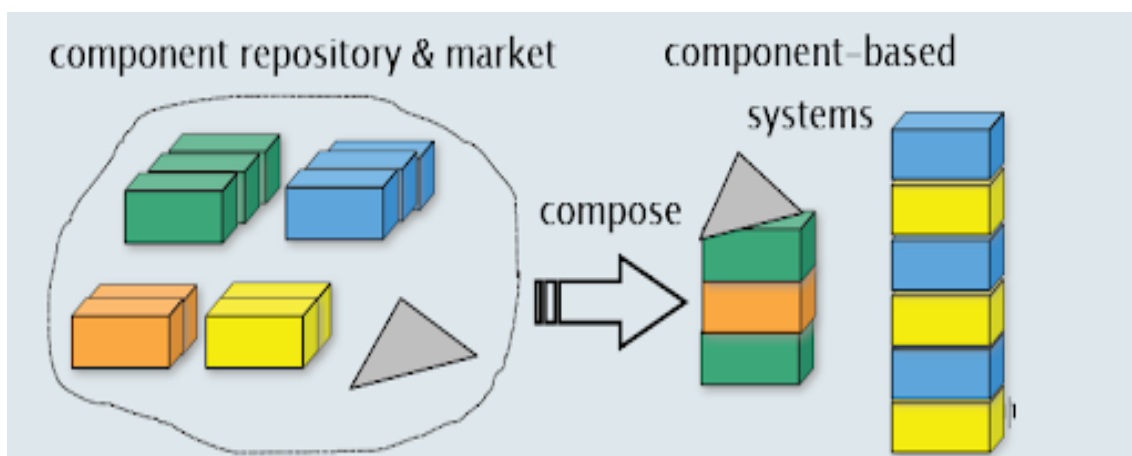


Figure 1: Concept of Component-based software engineering

3.2 Requirements

3.2.1 System Requirements

The system requirements for running the components are more in the software side than on the hardware. The hardware required is bare minimum for running a Java Application and a networking hardware. The hardware components are the server, client, peer, transmission medium, and connecting devices. The software requirements are:

- Java Development Kit (JDK)

It provides a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development. The JDK is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows[3].

- Tomcat Server

The Apache Tomcat software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. The Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket specifications are developed under the Java Community Process[1].

- MySQL Server

The MySQL Database software is a client/server based software which is often used to store data and retrieve data with the help of certain queries by implementing a certain back-end depending on the framework used for accessing the database.

- Eclipse IDE

The Eclipse IDE is one of the most famous Java IDE (Integrated Development Environment) which are mostly used to build and easily debug an application that is being developed. It allows us to get multiple language support which helps in debugging and finding errors in real-time. It also provides us with snippet support and many other features.

3.2.2 Functional Requirements

Component-based Software provides new opportunities for the development of secure, dependable and flexible systems. These components have their own functional interface for proper functioning and it hides the complexity of the component from the end-users. The several functional requirements for our websites are:

- **Stakeholders**

Stakeholder analysis is very important for a development project such as ours, since stakeholders are the people who show interest in our project and hence winning their support is important. The stakeholders for our system would be the students, developers, and organizations with software needs.

- **Actors and Goals**

Actors are the people or things (other applications, hardware, etc.) that will interact with our system to get something done. Each actor has a set of goals – tasks they need to get done using the system we are specifying.

- *Primary*

- * End User: create account, login to account, create profile, search components, download/preview components

- *Secondary*

- * Database Server: provides and stores data

- **Use Cases**

The use-cases of our systems are the end-users, developers and the administrator for the repository.

- **End-Users:** These are the clients and organizations which depend on the components for developing a system of their need

using the components available. Their support acts as a motivation for the developers who provide support for their components and the developers are responsible for the proper working, security and execution of their components.

- **Developers:** These are the people with proper technical knowledge and know about the working of the components and they are responsible for updating and making the components secure with time.
- **Administrator:** The administrator plays a vital role in authenticating each and every user for the proper functioning of the system, he has the privileges to verify the components and manage users.

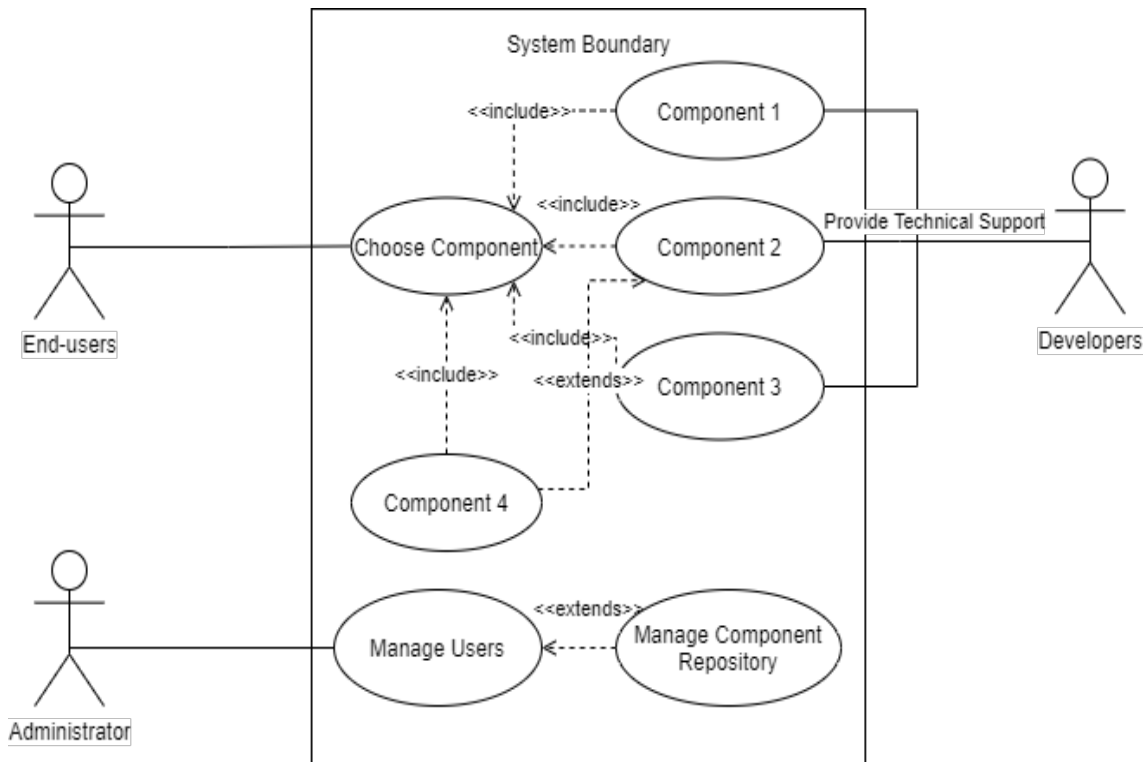


Figure 2: Use Case Diagram of the System

- Flow Chart Diagram** The flow chart diagram below will give you a proper understanding of the model of our system. The end-users have to select their preference of their base component and then they are given a set of choices of other comprising components which they can select or unselect by checking the check-boxes in the main dashboard interface. After the users are sure of their choices they can opt to preview the component user interface, that we're providing or just directly download the component as per their wish.

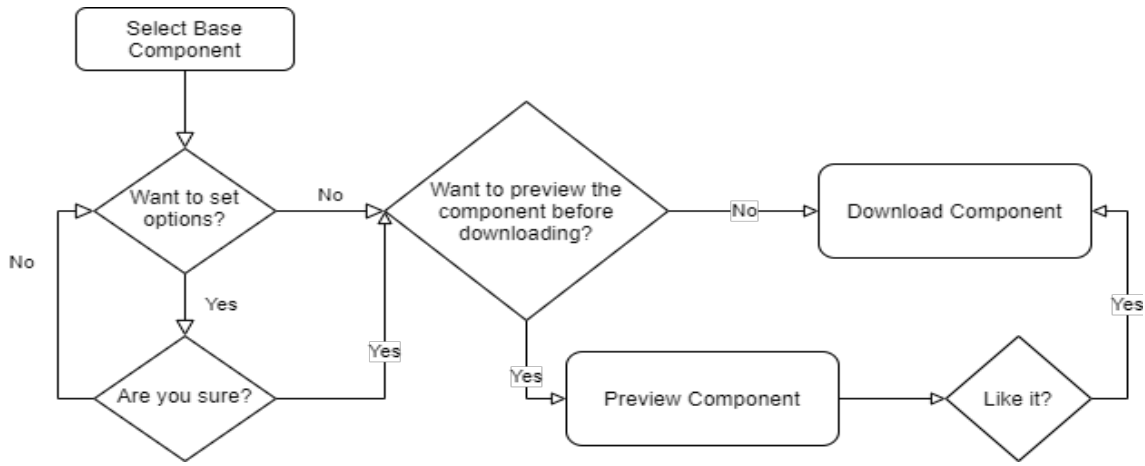


Figure 3: Flow Chart Diagram for the proper understanding of the system

3.3 Specifications

3.3.1 User Interface

User interface is an important part of any software and here we provide easy to use interface for the end users.

The end users can register to the system by providing minimal details and then login to the system using their username and password (provided in during registration). Users can anytime update their profile information

by using "update details" tab in website. Users can view their profile in "dashboard" tab, view administrator contact info in "contact us" tab and view usage of the website in "about us" tab provided in the website. Main use case of website is provided in "Home" tab. There users can select category/domain of website they want components for using a drop-down list. The components of selected category will appear below in form of check boxes. Users can select their desired components and use preview or download button respectively.

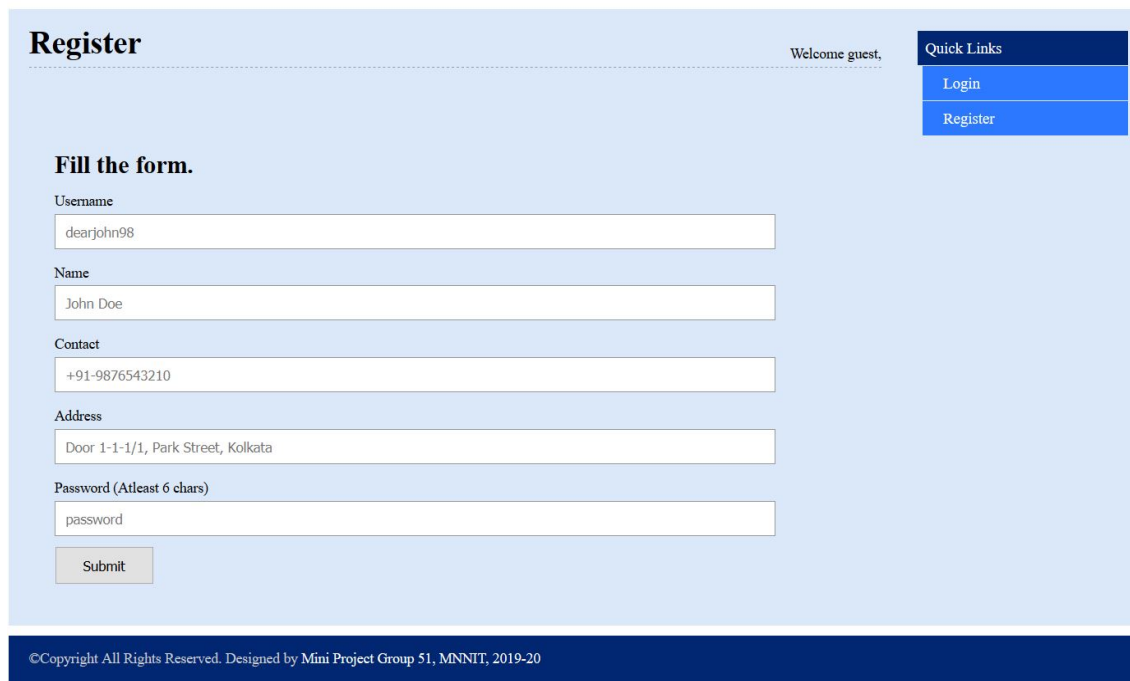
3.3.2 Preliminary Design

Let's consider a valid use case where a user wants to make a library management system. For reducing development time and cost the user wishes to apply component-based software development. The user comes to website in search of library management system components.

The interface interaction of the user with the website for this use case is depicted below -

- **Register**

The user provides minimal details required for registration if not registered. These details are used for creating an user account for authentication purposes.



The image shows a user registration form titled "Register". At the top right, there is a "Welcome guest," message and a "Quick Links" menu with "Login" and "Register" buttons. The form itself has a heading "Fill the form." and contains five input fields: "Username" (with "dearjohn98"), "Name" (with "John Doe"), "Contact" (with "+91-9876543210"), "Address" (with "Door 1-1-1/1, Park Street, Kolkata"), and "Password (Atleast 6 chars)" (with "password"). A "Submit" button is located at the bottom of the form. The footer of the page contains the text "©Copyright All Rights Reserved. Designed by Mini Project Group 51, MNNIT, 2019-20".

Register

Welcome guest,

Quick Links

- Login
- Register

Fill the form.

Username
dearjohn98

Name
John Doe

Contact
+91-9876543210

Address
Door 1-1-1/1, Park Street, Kolkata

Password (Atleast 6 chars)
password

Submit

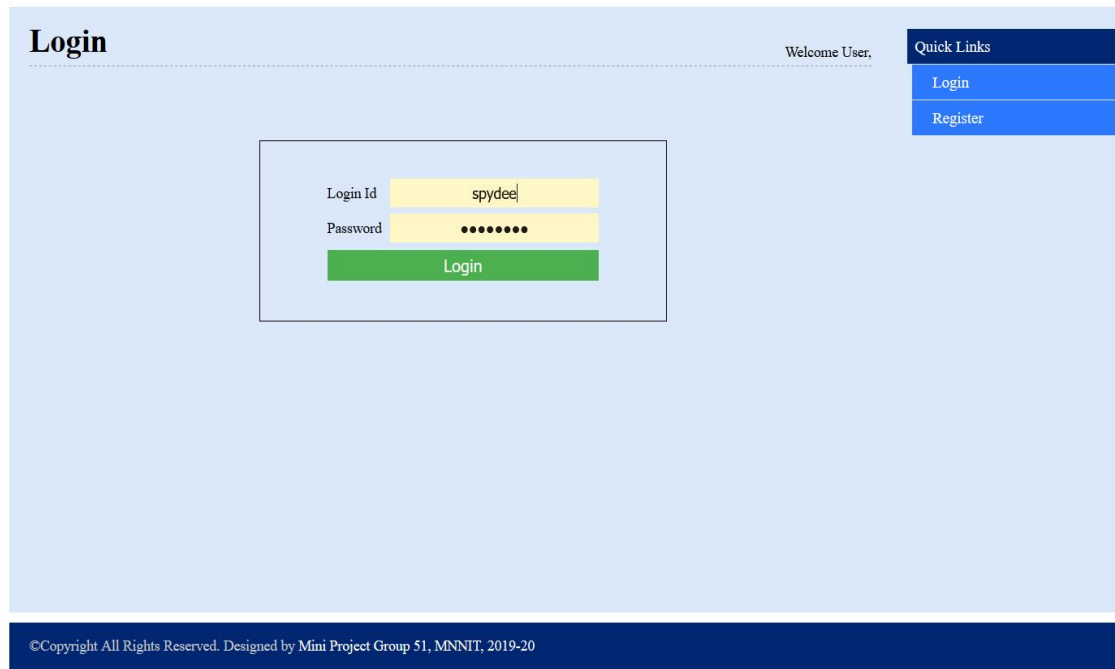
©Copyright All Rights Reserved. Designed by Mini Project Group 51, MNNIT, 2019-20

Figure 4: User registration screen

After successful registration user can login in to the system from login page.

- **Login**

The user provides username and password (entered at time of registration) in respective fields and clicks login button for logging in. These user input values are case sensitive and the user will only be able to login with the correct credentials.



The login screen features a light blue background. At the top left, the text "Login" is displayed in a bold, dark font. To the right, a "Welcome User," message is visible. In the center, there is a white rectangular box containing the login form. The form has two input fields: "Login Id" with the text "spydee|" and "Password" with masked characters "••••••••". Below these fields is a green "Login" button. On the right side of the page, there is a "Quick Links" section with a dark blue header and two blue buttons labeled "Login" and "Register". At the bottom of the page, a dark blue footer contains the copyright notice: "©Copyright All Rights Reserved. Designed by Mini Project Group 51, MNNIT, 2019-20".

Figure 5: Login screen

Validity of username and password is checked by system and if found valid user is redirected to users home page.

- **Home**

After successfully logging in, user is navigated to home page for category selection. User can next select the desired category from the drop-down list. In this page user can also navigate to various other tabs like - "dashboard", "update details", "about us" (to find usage of software), etc.

User can next select the desired components by selecting or unselecting the check boxes.

Component Selection

Welcome Spider Man,

Select a Category: Library Management System

Select Components:

Search Books

Show Books

Issue/Return Books

Add Books to the Library System

Edit Books in the Library System

☒

☒

☒

☒

☒

Preview Page

Download Components

Quick Links

Dashboard

Update Details

Logout

©Copyright All Rights Reserved. Designed by Mini Project Group 51, MNNIT, 2019-20

Figure 6: Home screen: component selection

User then selects button according to his/her desire, let's say user selects "preview components" button.

- **Preview Components**

By selecting "preview components" button the user is directed to new preview tab. This shows the user the user interface of the components he chose by checking the check-boxes.

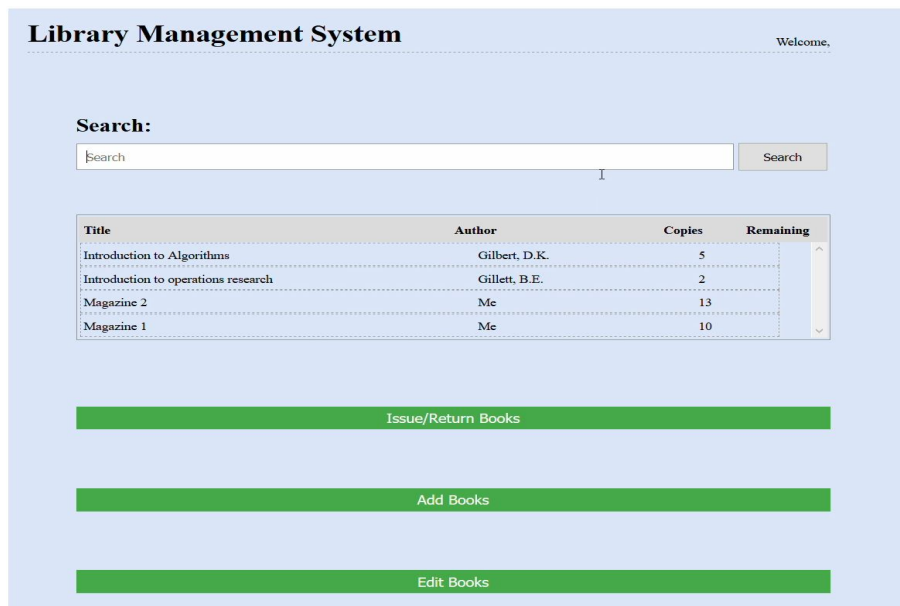


Figure 7: Preview screen

The user can then interact with the desired component.

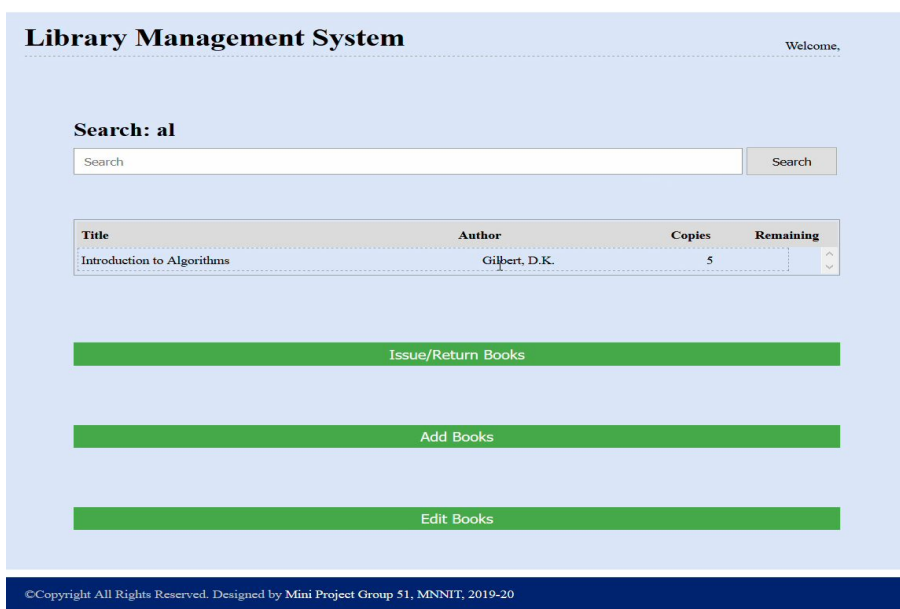


Figure 8: Preview screen: interaction with search component

Library Management System

Welcome,

Issue/Return

Fill the form.

An user can only issue a maximum of 3 books at a time.

Username

Spider Man - spydee

Books

Introduction to Algorithms - Gilbert, D.K.

Submit

Go Back to Preview Page

©Copyright All Rights Reserved. Designed by Mini Project Group 51, MNNIT, 2019-20

Figure 9: Preview screen: interaction with issue/return component

- User Effort Estimation

In the provided use case, the number of total clicks/keystrokes is 90 (avg).

- *User interface interaction*

number of clicks/keystrokes is 10 (avg).

Fraction = 0.11

- *Clerical data entry*

number of clicks/keystrokes is 80 (avg).

Fraction = 0.89

3.3.3 Object Interface

An interface is an overview of the actions/methods that an object can do, for example when you turn on a light switch, the light goes on, you don't care how, just that it does. In Object Oriented Programming, an Interface is an overview of all the functions that an object must have to be an object, i.e. enforcing certain properties on an object. An interface has a very similar syntax as that of a class definition. To implement a certain interface on a class object, we use the keyword 'implements'.

```
1 package{
2     public class Car implements Vehicle{
3         public void start_engine()
4         {
5             //code to write
6         }
7     }
8 }
```

The interface requires the programmer to create specific functions that are expected in an implementing class when it implements an Interface[2]. And hence this is how we can maintain certain standards for our components with the help of an interface.

Since our project is Java based web application, hence most of the data is stored in our database and therefore all the data is retrieve from the databases and stored in Data Access Objects (DAO). And these Data Access Objects receive their data from the Data Source which is a configuration file that is stored in the root directory of the package and the file is in XML format, which is easily understandable for the user to change their configurations according to their respective systems. The Java Database Connector (JDBC) is used to fetch data and order them according to the Data Access Object. The JDBC Template is used to order the data according to the data access objects. The directory of the package of the components are as follows:

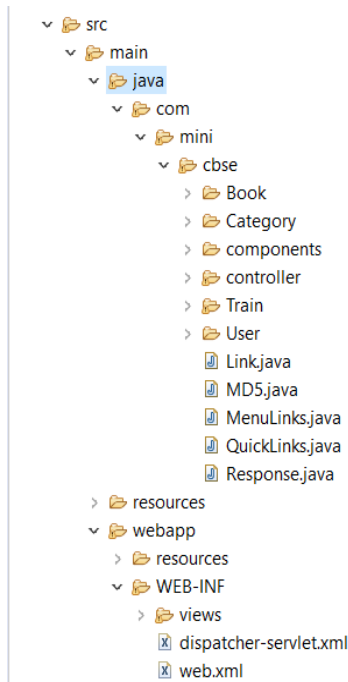


Figure 10: Directory View of the Component Source

Each of the components are present in the "cbse" directory inside the main package of "java.com.mini". Each of these components have at-least a Data Access Object, JDBC Template and a Mapper to map the data retrieve by JDBC Template to the variables in the Data Access Objects. The interface provided by each of these genesis components are as follows:

1. Book

This component is used for the library system. In the library system, this interface helps in adding a book, verifying the existence of the book in the library, issuing and returning of the book.

```
1 package com.mini.cbse.Book;  
2  
3 import java.util.List;  
4 import com.mini.cbse.Response;  
5 import javax.sql.DataSource;
```

```

6
7 public interface BookDAO{
8     public void setDataSource(DataSource ds);
9     public Response checkIfExists(Integer id);
10    public Response create(String name, String author,
        Integer total, Integer rem);
11    public Book getBook(Integer bookid);
12    public List<Book> listBooks();
13    public List<Book> searchBook(String query);
14    public Response delete(Integer id);
15    public Response update(Integer id, Integer total,
        Integer rem);
16    public Response issueBook(Integer id, String
        username, String on_counter);
17    public Response returnBook(Integer id, String
        username, String on_counter);
18    public Response toggleIssueReturn(Integer id, String
        username, String on_counter);
19    public List<Book> getIssuedBooksByUsername(String
        username);
20 }

```

2. Train

This component is used for the Railway system. In the Railway system, this interface helps in checking the price of the tickets, getting the train details for certain routes and making a reservation for a seat on the train.

```

1 package com.mini.cbse.Train;
2
3 import java.sql.Date;
4 import java.util.List;
5
6 import javax.sql.DataSource;
7
8 public interface TrainDAO {
9     public void setDataSource(DataSource ds);
10    public Train getTrainByTno(int tno);

```

```

11 public List<Train> listTrains();
12 public List<TrainSchedule> getTrainsByFromToStation(
    String from, String To);
13 public int getTicketCost(int train_no,int from_no,
    int to_no, String selectedClass);
14 public int getAvailableSeats(int train_no);
15 public int makeReservation(String username,String
    name,TrainSchedule ts,
16     String date,String selectedClass,int adult,int
    child,int total_cost);
17 }

```

3.3.4 Formatting queries

The systems or components are provided with a user interface which are present in the "views" directory and are controlled using the controllers present in the "controller" directory. Since, we're using the Spring Framework, hence we can use views and present our models using our controllers. The controller is responsible for presenting the output and process the given data by the database and the user input. The URL's are mapped with a certain view hence can be changed according to the preference of the user. The queries are hard-coded and are present inside the class files of the components. The controller is the main important part of forming a system. For example,

```

1 package com.mini.cbse.controller;
2 @Controller
3 public class LibraryController {
4
5     @RequestMapping(value="/books", method=RequestMethod.GET
6         )
7     public ModelAndView showBooks(HttpServletRequest request
8         ,@RequestParam(value = "search", required = false,
9         defaultValue = "") String search) {
10         Component comp = new ShowBooksComp();
11         ModelAndView mv = new ModelAndView();

```

```

9      Map<String, Object> parameter = new HashMap<String,
      Object> ();
10      parameter.put("search", search);
11      comp.doGetAction(request, parameter);
12      mv.setViewName(comp.getViewName());
13      mv.addAllObjects(comp.getModelMap());
14      String catname = (String) request.getSession().
      getAttribute("category_name");
15      mv.addObject("category", catname);
16      return mv;
17  }
18 }

```

Here, this is the part of the controller for the library system, and this controller has a URL Mapping with the URL path `"/books"` mapped with a view as `"constructed_page"`.

The queries are declared inside the JDBC Template of the components as:

```

1  private String SELECT_ID = "SELECT * from books where id
   = ?";
2  private String INSERT_BOOK = "INSERT into books (
   book_name, book_author, total_copies, rem_copies)
   values (?, ?, ?, ?)";
3  private String SELECT_ALL_BOOKS = "SELECT * from books";
4  private String SEARCH_BOOKS = "SELECT * from books where
   UPPER(book_name) LIKE UPPER(?) or UPPER(book_author)
   LIKE UPPER(?)";
5  private String DELETE_BOOK_ID = "DELETE from books where
   id = ?";

```

3.4 Domain Analysis

The major parts of the domain of our project includes:

1. Library Management System

The library management systems are needed in various institutions to maintain all the data and the count of the books with their issuing and returning details. Having a system makes it flexible to add new book records or change the count or remove several issues which would be very hard when data is maintained physically. The primary actors are the students and the admin or registrar.

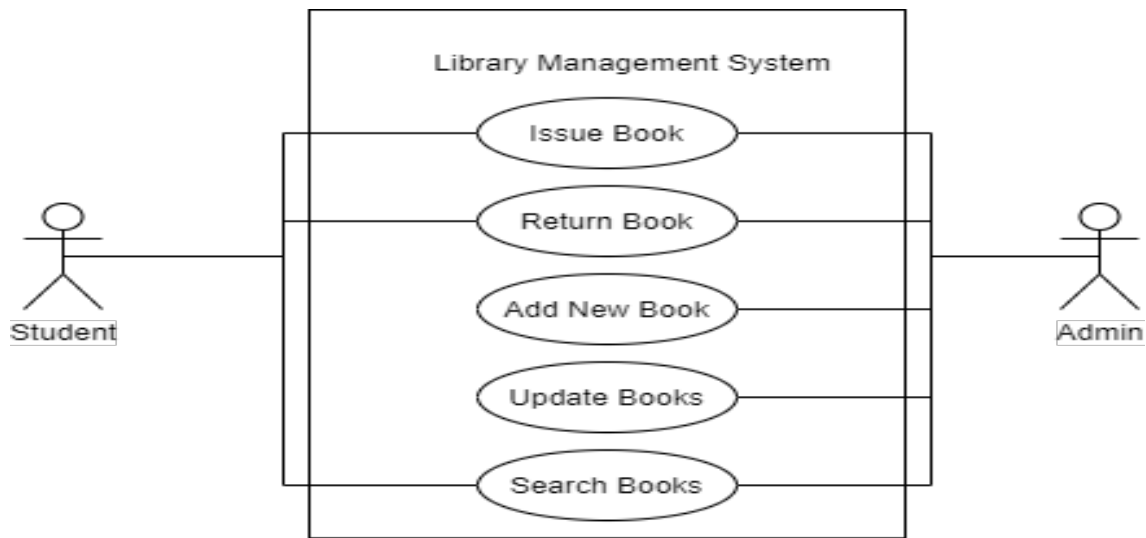


Figure 11: Use case diagram of Library Management System

2. Railway Management System

The railway management systems are needed in various countries to maintain all the data and the count of the reservations and income with their user details and train details. Having a system makes it flexible to add a new train record or the user details or remove several issues which would be very hard when data is maintained physically. The primary actors are the general people and the admin. These systems need to be secure and reliable due to their implementation as large-scale systems.



Figure 12: Use case diagram of Railway Management System

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Hence through this work, we can conclude that Component-Based Software Engineering is essential for reuse of components and also provides flexibility in developing a complex and good quality software applications over a short period of time. We observed that the components are created using Object Oriented Programming and hence can be very detailed and specific.

4.2 Future Work

We plan for more efficient back-end for classification and retrieval of required components from the repository. We would also like to provide design components as well as source code components (like functions, classes, libraries, ...) in future.

We would like to extend to more technologies like- machine learning (for efficient searching and retrieval of components by user provided keywords), block-chain (for certification and validation of components), distributed databases, etc.

We plan to provide more features with component insertions and component management for individual users.

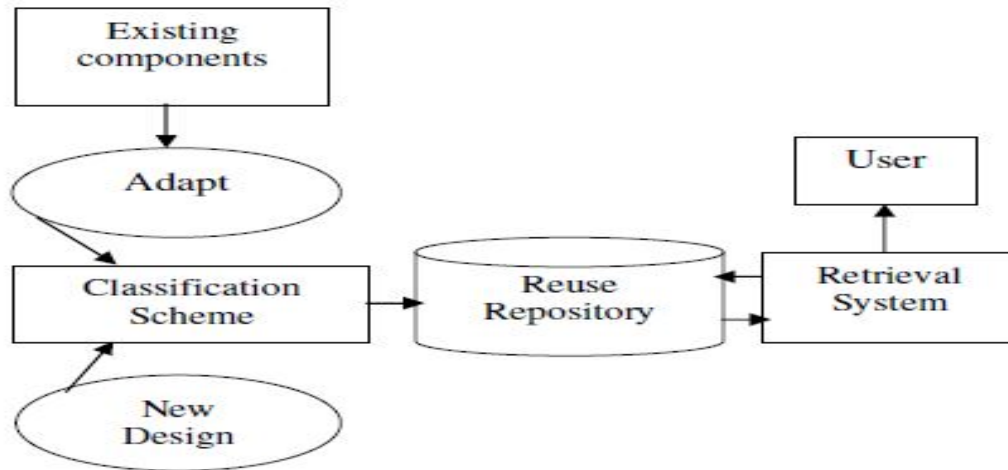


Figure 13: Proposed System Architecture[12]

References

- [1] Apache Tomcat. <https://tomcat.apache.org/>. Accessed: 2020-05-07.
- [2] Interfaces. <https://www.cs.utah.edu/~germain/PPS/Topics/interfaces.html>. Accessed: 2020-05-07.
- [3] JDK 14 Documentation. <https://docs.oracle.com/en/java/javase/14/>. Accessed: 2020-05-07.
- [4] potential customer. <https://stackoverflow.com/questions/2668694/software-engineering-component-repository-tool>. Accessed: 2020-05-07.
- [5] Research Areas of the Software Engineering Group. Online Available: <http://www.cs.uni-paderborn.de/en/research-group/software-engineering/research/research-ares.html>. Accessed: 2020-05-07.
- [6] ADEKOLA, O., IDOWU, S., AND ADEBAYO, A. Component based software engineering in student management system domain: A development for reuse approach. *International Journal of Software Engineering and Its Applications* 10 (09 2016), 149–162.
- [7] AOYAMA, M. New age of software development: How component-based software engineering changes the way of software development?
- [8] BRERETON, P., AND BUDGEN, D. Component-based systems: A classification of issues. *IEEE Computer* 33 (2000), 54–62.

- [9] BROOKS, A., KAUPP, T., MAKARENKO, A., WILLIAMS, S., AND OREBÄCK, A. *Orca: A Component Model and Repository*, vol. 30. 04 2007, pp. 231–251.
- [10] KHAN, A., QAYYUM, N.-U., AND KHAN, U. An improved model for component based software development. *Software Engineering* 2162-8408 2 (10 2012), 138–146.
- [11] KUNG-KIU LAU, AND ZHENG WANG. A taxonomy of software component models. In *31st EUROMICRO Conference on Software Engineering and Advanced Applications* (2005), pp. 88–95.
- [12] P.NIRANJAN, D. C. G. R. A mock- up tool for software component reuse repository.