SOFTWARE ENGINEERING LAB

PROJECT REPORT ON

SOFTWARE COMPONENT AND CATALOGUING SOFTWARE

Submitted By:

P. Chanukya Kumar

121CS0692

Submitted To:

Department of Computer Science and Engineering

Department of Computer Science and Engineering NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

APRIL, 2024

ABSTRACT

This report provides an overview of the Software component and cataloguing software, including its features, UI, backend, benefits and limitations. This software allows users to search, view details, manage metadata, control versions, set access permissions, collaborate, integrate, and access analytics and reporting functionalities. The software is designed to to organize, manage, and streamline the use of software components efficiently in the development processes.

Keywords:

1. SCCS
2. SRS
3. RAD Model
4. UML Diagrams
5. Implementation
6. Design
7. Database
8. UI
9. Backend

TABLE OF CONTENTS

List of Figures

List of Tables

1. LIST OF ABBREVIATIONS

SCCS Software Component Cataloging System

Stats Statistics

SRS Software Requirements Specification

RAD Rapid Application Development

SDLC Software Development Life Cycle

UML Unified Modeling Language Diagrams

UI User Interface

2. INTRODUCTION

2.1. Background

Modern software development, marked by agile methodologies and microservices architectures, emphasizes the need for efficient software component management. Traditional approaches struggle with version control, compatibility, and collaboration across teams. The rise of component-based development and open-source communities underscores the importance of effective cataloguing and organization of software components. This project addresses these challenges by developing a dedicated software component and cataloguing system.

2.2. Motivation

The motivation behind developing a software component and cataloging software is to streamline the organization and management of software components. This includes enhancing accessibility, facilitating version control, and promoting collaboration among users to maximize efficiency and productivity in software development processes.

2.3. Objectives

The objectives of the software component and cataloguing software project are:

1. Streamline the organization and categorization of software components.
2. Facilitate easy access and retrieval of software components.
3. Enhance version control and management of software component versions.
4. Promote collaboration among users for effective component sharing and utilization.

2.4. Problem statement

The challenge is to develop a robust software component and cataloging system

that can efficiently organize, manage, and track software components across projects and teams. This involves addressing issues related to versioning, metadata management, access control, and integration with development workflows.

2.5. Scope of Project

The scope of the software component and cataloguing software includes:

1. Creation of a centralized repository for storing and cataloguing software components.
2. Implementation of version control mechanisms to track changes and updates to components.
3. Integration with development tools and environments for seamless component usage.
4. User-friendly interface for easy navigation, search, and retrieval of components.
5. Access control features to regulate user permissions and roles within the system.
6. Collaboration functionalities to promote teamwork and knowledge sharing among users.

3. LITERATURE REVIEW

Literature Review: Software Component and Cataloging Software

Software component and cataloging software are essential tools in modern software development, facilitating the organization and management of software components across projects and teams. While there is a wide range of software available, including basic calendar apps to advanced project management systems, this review focuses on research related to software component and cataloging software specificallydesigned for executives.

One study highlights the impact of software component and cataloging software on productivity and job satisfaction among executives. Executives using such software reported higher levels of productivity and satisfaction, thanks to improved scheduling, prioritization of work, and reduced time spent on administrative tasks.

Another study delves into the effectiveness of software component and cataloging software tailored for project managers. This research demonstrates how such software can enhance time management skills, mitigate project delays, and ultimately lead to higher project success rates. The findings suggest that integrating software component and cataloging tools into project management practices can significantly improve outcomes and efficiency.

4. REQUIREMENT ANALYSIS

4.1 Introduction:

4.1.1 Purpose:

The Software Component and Catalogue Software are designed to streamline the process of managing software components, dependencies, and versions within a development environment. This software will assist developers in efficiently organizing, cataloging, and retrieving software components as needed for their projects. The purpose of this document is to specify the requirements for the Software Component and Catalogue Software.

4.1.2 Document Conventions:

There are no typographical conventions for this SRS document. The important parts in the document will be underlined. The higher priority functional requirements will be in bold text.

4.1.3 Intended Audience and Reading Suggestions:

This document is intended for developers, testers, and maintainers of the Software Component and Catalogue Software. All stakeholders involved in the project will have access to this document.

4.1.4 Product Scope:

The Software Component and Catalogue Software should provide a centralized platform for managing software components, their versions, dependencies, and related documentation. It should allow developers to easily search for, add, update, and remove components from the catalogue. Additionally, it should facilitate version control and tracking of component usage across projects.

4.1.4 Overview:

This document will primarily focus on the following aspects of the Software Component and CatalogueSoftware:

i. Overall Description

ii. Specific Requirements

Overall Description will outline the major components of the system, their

interconnections, and the general functionality of the software. Specific Requirements will detail the roles of actors within thesystem, their functions, and any constraints faced by the software.

4.2 Overall Description:

4.2.1 Product Perspective:

The Software Component and Catalogue Software will serve as a centralized platform for managing software components, dependencies, versions, and related documentation within a development environment. It aims to streamline the process of software development by providing developers with easy access to necessary components and ensuring version control and compatibility.

4.2.2 Product Functions:

- **User Authentication:** The users have to log in by giving their corresponding id and password for authentication purpose. After proper authentication they can use the other facilities of the system.
- **Component Management:** Allows users to add, update, and remove software components from the catalogue.
- **Version Control:** Tracks and manages different versions of software components, ensuring compatibility and enabling rollback if needed.
- **Dependency Tracking:** Manages dependencies between software components, ensuring that required dependencies are available and compatible.
- **Documentation Management:** Stores and organizes documentation related to software components, facilitating knowledge sharing and reference.
- **Search and Retrieval:** Enables users to search for specific software components based on criteria such as name, version, or tags, and retrieve them for use in projects.
- **Collaboration Tools:** Provides features for collaboration, such as sharing components with team members, commenting on components, and discussing changes.
- **Reporting and Analytics:** Generates reports and provides analytics on component usage, version history, and dependencies, aiding in decision-making and optimization.

4.2.3 User Class and Characteristics:

- There are several categories of users for the Software Component and Catalogue Software:
- Developers (Users): Individuals responsible for building and maintaining software projects. They interact with the catalogue to access and integrate software

components into their projects.

- System Administrators (Adm: Users with administrative privileges who manage the catalogue, configure settings, and oversee user access and permissions.
- Project Managers: Users responsible for planning and coordinating software development projects. They utilize the catalogue to track component usage, monitor progress, and ensure project timelines.
- Quality Assurance (QA) Engineers: Users who test software components for functionality, compatibility, and performance. They may use the catalogue to access specific versions of components for testing purposes.

4.2.4 General Constraints:

- Users must have access to a computer or laptop with an internet connection to use the
- Software Component and Catalogue Software effectively.
- The software will store information about software components, versions, and dependencies in a database.
- Only authorized users, such as developers and system administrators, can make changes to the software catalogue using their respective usernames and passwords.

4.2.5 User Documentation:

Although the software is designed to be intuitive and user-friendly, a PDF document will be provided with the software to explain its features, functionalities, and usage guidelines.

4.2.6 Assumptions and Dependencies:

Users are assumed to have basic knowledge of computer usage, including familiarity with input devices like mouse and keyboard, and understanding of standard dialogues and actions within the software.

A stable network connection is required for the Software Component and Catalogue Software to function properly, enabling communication between client devices and the database server.

Data about software components, versions, and dependencies will be inputted into the system during installation or setup.

4.3 Specific Requirements:

4.3.1 External Interfaces:

Hardware Interface:

The Software Component and Catalogue Software will run on standard desktop or laptop computers with input devices such as a mouse and keyboard, along with a stable power supply.

Software Interface:

The software requires an Operating System (preferably Windows) and a MySQL database server for data storage and retrieval.

Communication Interface:

During registration or setup, a representative of the main administrators will handle the communication interface to connect the software to the internet for remote access to database servers.

User Interface: The software should have a user-friendly graphical user interface (GUI) for developers, system administrators, and other users to interact with. It should support actions such as adding, updating, and removing software components, managing versions and dependencies, and accessing documentation. Notifications, reporting features, and collaboration tools should also be integrated into the GUI for efficient software management.

4.3.2 Functional Requirements:

User Authentication(): The system should ensure the security and privacy of user data.

User Registration(): This task will be done by the representative of the main administrators. They can enter the required details along with initial password for them which they can change later.

Component Management(): Allow users to add new software components to the catalogue with detailed information such as name, version, description, and dependencies. Enable users to update existing components by modifying version details, descriptions, or dependencies. Provide the ability to remove obsolete or deprecated components from the catalogue.

Version Control(): Implement version control mechanisms to track different versions of

software components. Support version comparison and rollback functionalities to revert to previous versions if necessary. Ensure that users can easily identify the latest stable version of each component.

Dependency Tracking(): Manage dependencies between software components by recording and displaying dependency information.
Notify users of any conflicts or compatibility issues between components and their dependencies.
Allow users to resolve dependency conflicts or update dependencies as needed.

Documentation Management(): Store and organize documentation related to software components within the catalogue.
Enable users to upload, view, and download documentation files such as user manuals, API references, release notes, and installation guides.
Provide version-specific documentation to align with different component versions.

Search and Retrieval(): Implement a search functionality that allows users to search for softwarecomponents based on various criteria such as name, version, tags, or description.
Enable users to filter search results and sort components based on relevance or alphabetical order.Support quick retrieval of components for use in software development projects.

Collaboration Tools(): Facilitate collaboration among team members by allowing them to share feedback, comments, and suggestions on software components. Provide a discussion forum or commenting feature within the catalogue platform for communication and collaboration.
Allow users to collaborate on component updates, bug fixes, and enhancements.

Reporting and Analytics(): Generate reports on component usage, downloads, and popularity within the catalogue. Provide analytics on trends related to component adoption, version preferences, and usage patterns.
Offer insights into the performance and impact of software components on development projects.

Integration(): Integrate with other development tools and platforms such as version control systems, project management software, and continuous integration/continuous deployment (CI/CD) pipelines. Ensure seamless data exchange and compatibility with external tools to streamline software development workflows.
Support integration with software repositories and package managers for automated component updates and deployments.

These functional requirements are crucial for ensuring effective management, tracking, and utilization of software components within the catalogue software. They contribute to improved collaboration, productivity, and decision-making in software development processes.

4.3.3 Performance Requirements:

Some common performance requirements that are essential for a Software Component and Catalogue Software, to be effective are:

Speed: The Software Component and Catalogue Software should perform operations such as adding, updating, and retrieving software components swiftly to minimize downtime and increase productivity. Response times for search queries, version comparisons, and dependency checks should be optimized for efficient usage.

Accuracy: Ensure accurate tracking of version history, dependencies, and documentation related to software components within the catalogue. Avoid data discrepancies or inconsistencies that could lead to compatibility issues or deployment errors.

Usability: Provide a user-friendly interface for managing software components, dependencies, and versions within the catalogue software. Make navigation intuitive and include helpful tooltips or guides for new users to understand the functionality of the software.

Customization: Allow users to customize settings and preferences within the catalogue software to align with their specific needs and workflows. Customizable features enhance user satisfaction and adoption by providing flexibility in usage.

Integration: Support integration with version control systems, project management tools, and development environments to facilitate seamless collaboration and workflow automation. Ensure compatibility with common software repositories and package managers for streamlined software development processes.

Security: The system should be secure and protect user's data from unauthorized access or theft. Security breaches can result in data loss, privacy violations, and damage to the system's reputation.

Scalability: The system should be able to handle a growing number of users and data as the organization grows. A scalable system can save time and money by reducing the need to migrate to a new system as the organization expands.

4.3.4 Logical Database:

For a Software Component and Catalogue Software the logical database requirements should include the following:

Component Table:

This table will contain information about each software component, including its name, version, description, dependencies, and documentation links.
Used for cataloging and organizing software components within the catalogue software.

Version History Table:

This table will track the version history of software components, recording details such as version number, release date, changelog, and associated component ID.
Used for version control and tracking changes to software components over time.

Dependency Table:

This table will manage dependencies between software components, storing information about required dependencies, compatibility status, and version constraints.
Used to ensure that software components are compatible and can be integrated seamlessly.

Documentation Table:

This table will store documentation related to software components, including user manuals, APIreferences, release notes, and installation guides.
Used for easy access and retrieval of documentation within the catalogue software.

Search Index Table:

This table will contain indexed data for quick search and retrieval of software components based oncriteria such as name, version, tags, and descriptions.
Used to enhance search functionality and improve user experience when searching for components.

User Table:

This table will contain information about users interacting with the catalogue software, including theirname, email, username, password, and role (developer, admin, etc.).
Used for user authentication, access control, and personalized experiences within the software.

Settings Table:

This table will store user-specific settings and preferences, such as display options, notifications, anddefault views.
Used to provide customized experiences and enhance user satisfaction.

Activity Log Table:
This table will log user activities within the catalogue software, capturing details such as user ID,timestamp, action performed, and affected component or project.
Used for audit trails, tracking user actions, and monitoring system activity.

Overall, the logical database requirements for the Software Component and Catalogue Software are designed to efficiently manage and store data related to software components, versions, dependencies, documentation, user information, settings, and activity logs. These tables facilitate effective organization, retrieval, and management of information within the catalogue software.

4.3.5 Design and Implementation:

The Software Component and Catalogue Software should be designed using a suitable programming language like Java or Python to ensure platform independence. MySQL should be utilized for efficient database management to store and retrieve software component data effectively.

The software should be compatible with desktop computers meeting minimum specifications, including 4GB RAM, 128GB hard disk space, and a processor speed of at least 2.5GHz.
Compatibility with common operating systems such as Windows 10 should also be ensured.

The user interface should be designed to be user-friendly, intuitive, and easy to navigate, taking into account that users may have varying levels of experience with computer usage. It should include tooltips, help guides, and contextual information to assist users in understanding and utilizing the software effectively.

To ensure data security and privacy, appropriate authentication and access control mechanisms should be implemented, along with encryption techniques to protect sensitive information stored inthe database and during data transmission.

The software should be designed with modularity and extensibility in mind, allowing for the addition of new features or integration with other software systems in the future. This can be achieved by using design patterns and architecture principles that support scalability and flexibility in software development.

Performance optimization should be a key consideration, ensuring that the software can handle multiple requests and transactions without causing delays or disruptions. Standard protocols for email communication, such as SMTP, POP3, and IMAP, should be utilized for efficient email communication within the software.

4.3.5 Software System Attributes:

The Software System Attributes include:

Availability: The Software Component and Catalogue Software should be available to users whenever they need it, 24/7, to ensure continuous access and productivity.

Reliability: The software should be reliable, meaning that it should not crash or lose data during normal usage, ensuring data integrity and consistent performance.

Maintainability: The software should be easy to maintain and update over time, with clear documentation, modular design, and version control mechanisms in place.

Usability: The software should be user-friendly and easy to use, particularly for users with varyinglevels of experience, ensuring a smooth and intuitive user experience.

Scalability: The software should be able to handle a growing number of users and software components over time, adapting to increased demand and workload without performance degradation.

Security: The software should be secure, especially considering that it will handle sensitive data such as software components, dependencies, and user information. Robust authentication, access control, and encryption measures should be implemented.

Performance: The software should perform well, particularly when managing and organizing software components, conducting searches, and generating reports. It should handle operations efficiently and provide quick responses to user actions.

Portability: The software should be portable, meaning that it should be able to run on different types of computer systems and platforms without compatibility issues, ensuring flexibility and accessibility across various environments.

4.3.5 Business Rules:

The Software Component and Catalogue Software should enforce business rules such as requiring essential information for each added software component, maintaining version control with unique version numbers and detailed version history, managing dependencies to ensure compatibility and provide notifications for conflicts, including comprehensive documentation for components, enabling users to search and retrieve components based on various criteria, restricting access control to authorized users for data integrity and security, facilitating version comparisons and rollback options, generating notification alerts for new additions or critical updates, implementing an approval process for new component additions, enabling feedback and reviews for informed decisions, and ensuring secure storage and protection of all data from unauthorized access or manipulation, ultimately enhancing the efficiency, organization, and usability of software components within the catalogue software.

5. METHODOLOGY

5.1. Software Development Approach (SDLC)

Software Component and Catalogue Software Development Approach
The Software Component and Catalogue Software will be developed using the Agile methodology, specifically the Scrum framework. This approach is chosen for its iterative and collaborative nature, allowing for flexibility, adaptability, and frequent feedback cycles throughout the development process. The steps involved in the development process are as follows:

1. Requirements Gathering: Collaborate with stakeholders to gather requirements for the software component and catalogue software, including functionalities, user interface preferences, and integration needs..

2. Product Backlog Creation: Create a product backlog containing all the features, tasks, and user stories identified during the requirements gathering phase.

3. Sprint Planning: Break down the product backlog into manageable tasks and prioritize them f or implementation during each sprint.

4. Iterative Development (Sprints): Conduct iterative development in sprints, typically 2-4 weeks long, where development teams work on implementing and testing specific features from the product backlog.

5. Daily Stand-up Meetings: Hold daily stand-up meetings to discuss progress, address any impediments, and ensure alignment within the development team.

6. Sprint Review and Retrospective: At the end of each sprint, conduct a sprint review to demonstrate completed features to stakeholders and gather feedback. Also, conduct a sprint retrospective to reflect on the sprint's successes and areas for improvement.

7. Continuous Integration and Testing: Implement continuous integration practices to ensure that code changes are integrated and tested regularly, maintaining code quality and reducing integration issues.

8. User Acceptance Testing (UAT): Conduct user acceptance testing with stakeholders and end-users to validate that the software components meet their requirements and expectations.

9. Release and Deployment: Release software components and catalogue software

versions based on the feedback and approvals received during the sprint reviews and user acceptance testing.

10. **Monitoring and Maintenance:** Monitor the performance and usage of the software components and catalogue software post-deployment, addressing any issues or enhancements identified through user feedback or monitoring tools.

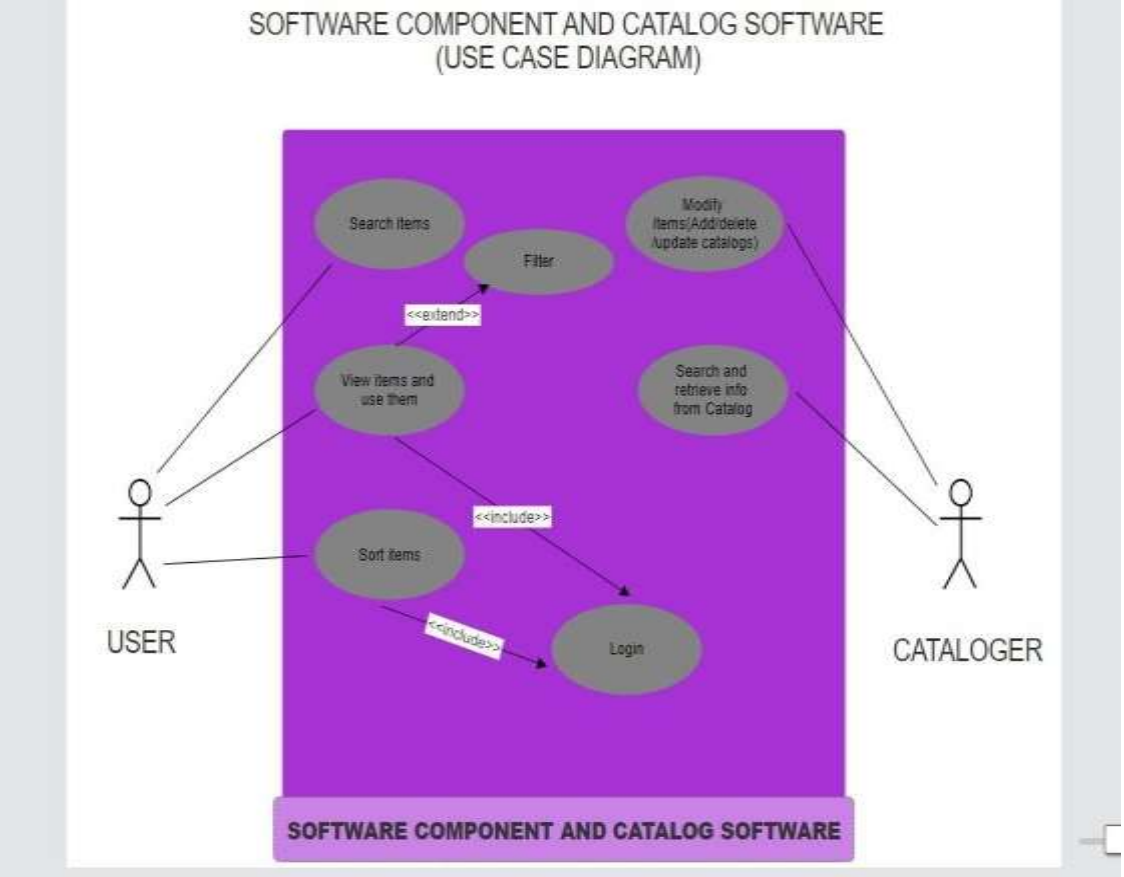
By adopting the **Agile methodology with the Scrum framework for software component and catalogue software development**, we can ensure a collaborative, iterative, and customer-centric approach, delivering high-quality software that meets user needs and adapts to changing requirements effectively.

6. SYSTEM ARCHITECTURE AND METHODOLOGY

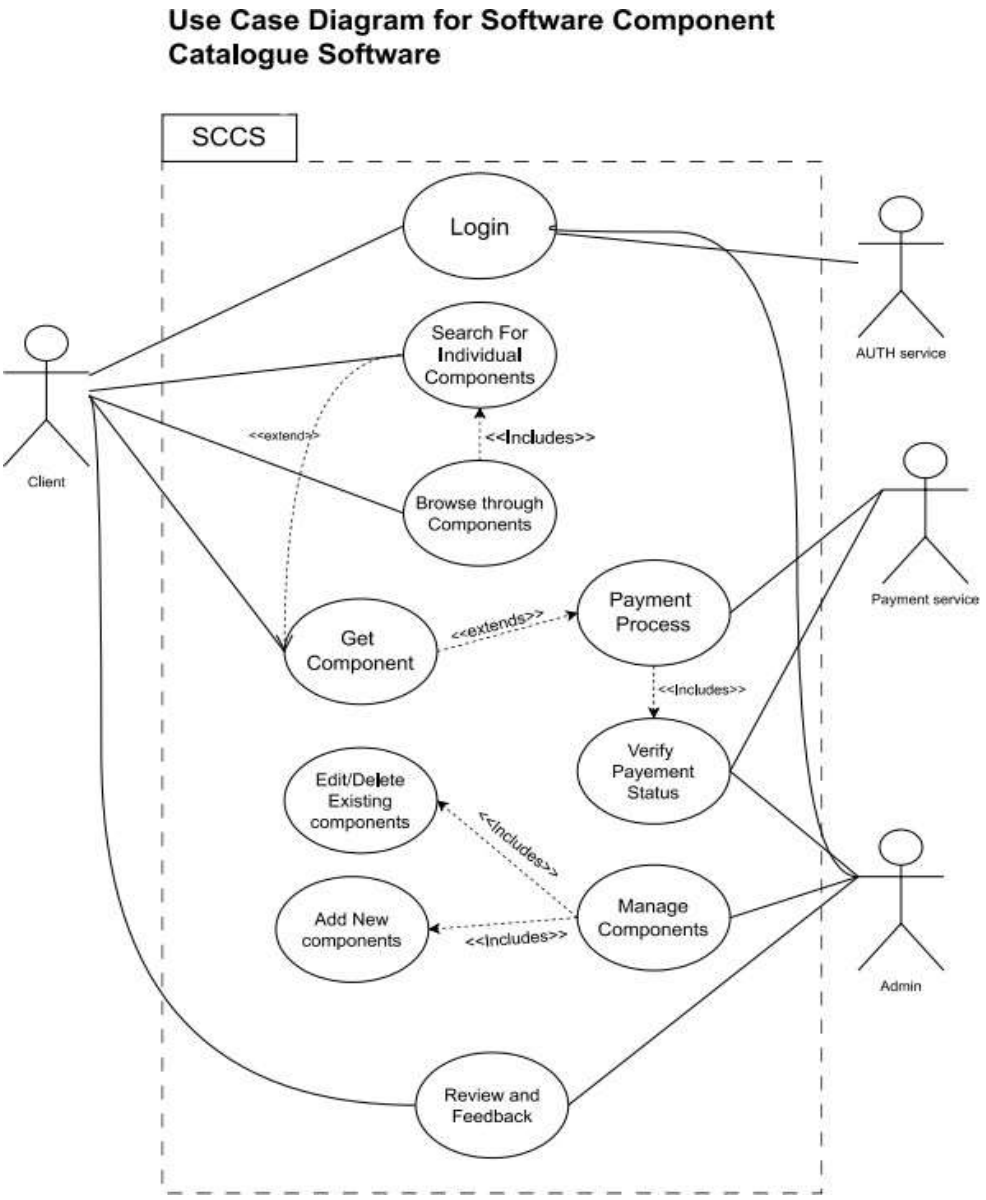
This section includes the theoretical background for the project, and description of the architecture and algorithm used in the project. Also, if possible, include the UML diagrams (use case, activity) and explain them in brief if you can.

6.1. Use Case Diagram

SIMPLE USE CASE DIAGRAM.

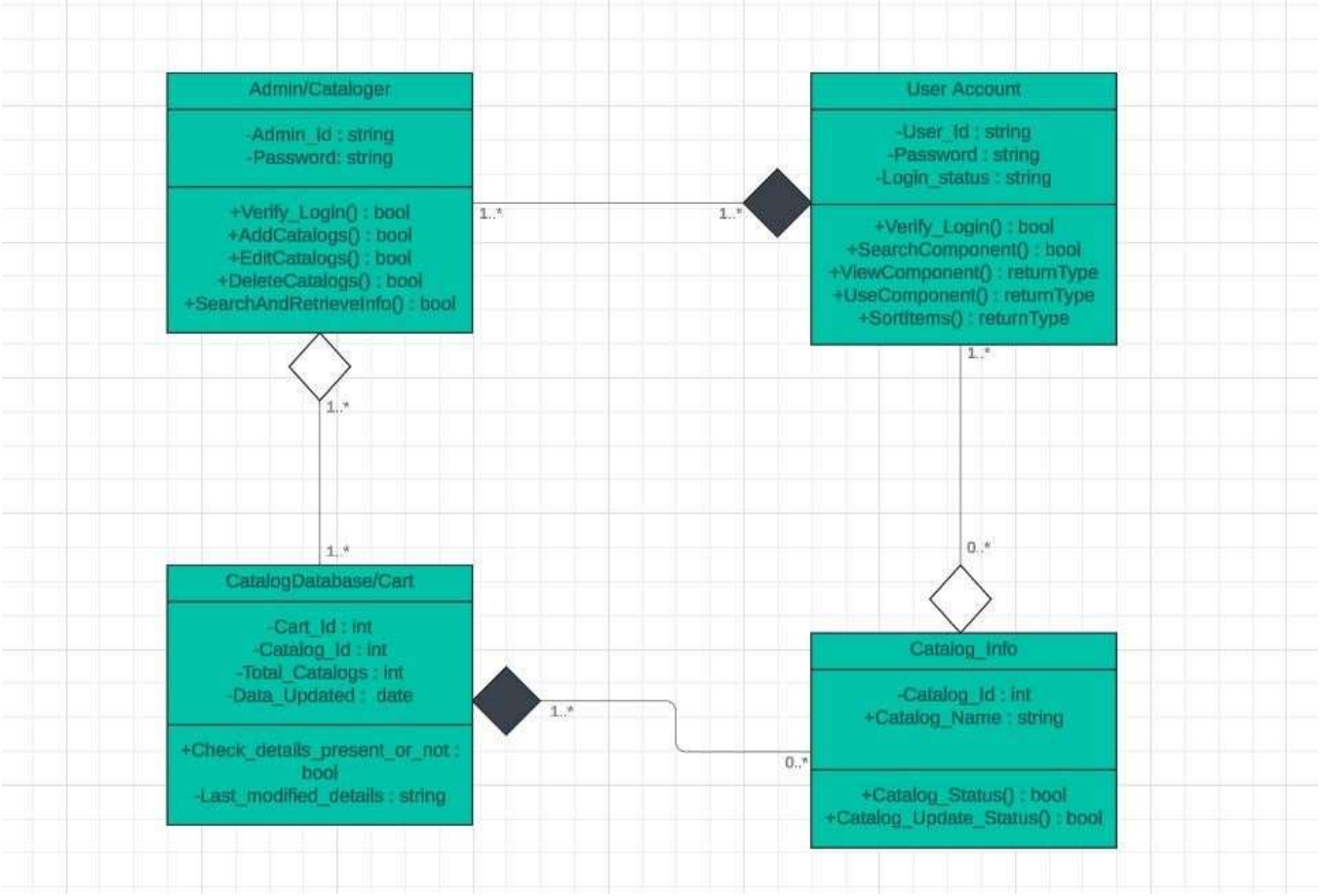


USECASE DIAGRAM WITH MULTIPLE ACTORS

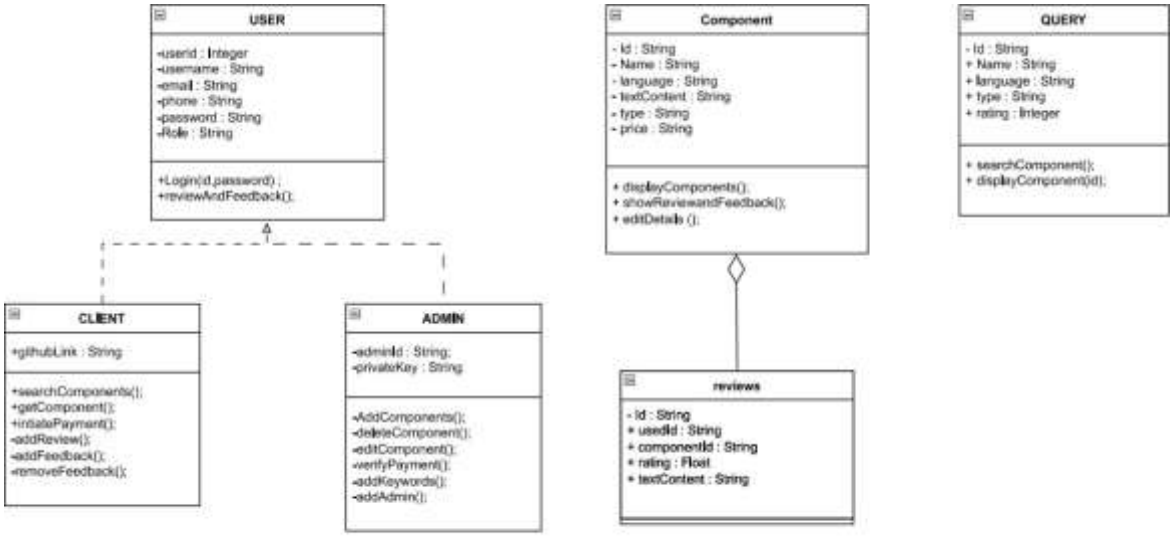


6.2. Class Diagram

SIMPLE CLASS DIAGRAM

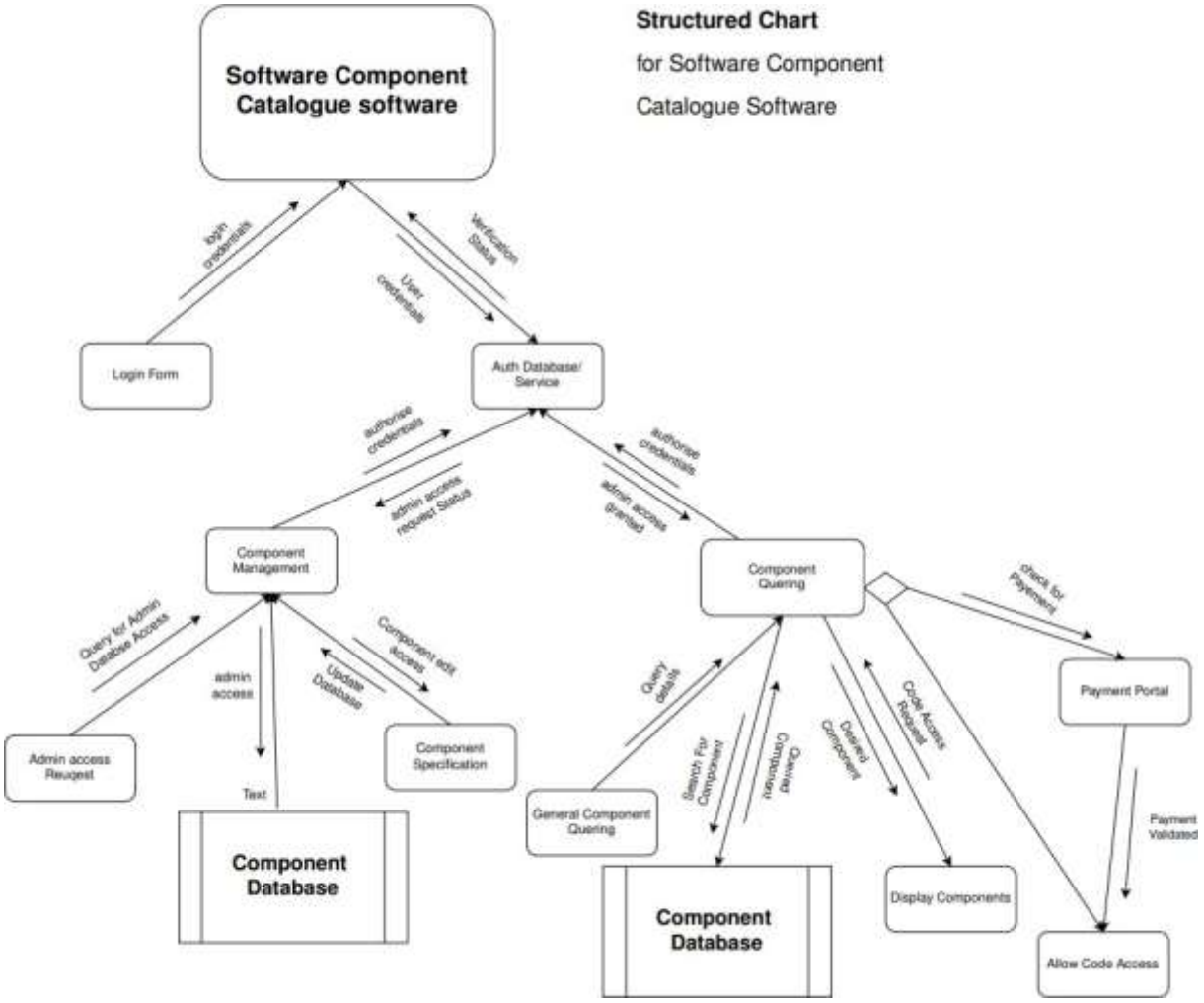


CLASS DIAGRAM WITH MULTIPLE CLASSES IN DETAIL

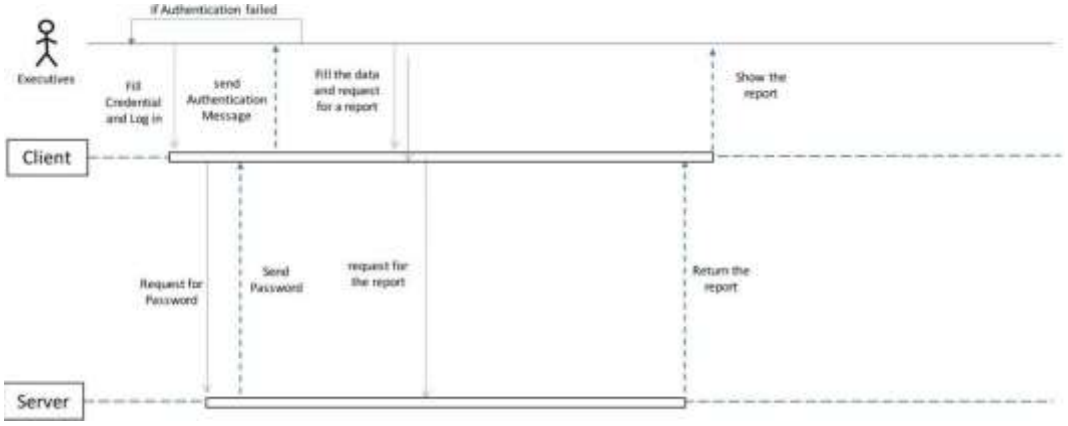


Class Diagram for Software Component Catalogue Software

6.3. STRUCTURE CHART



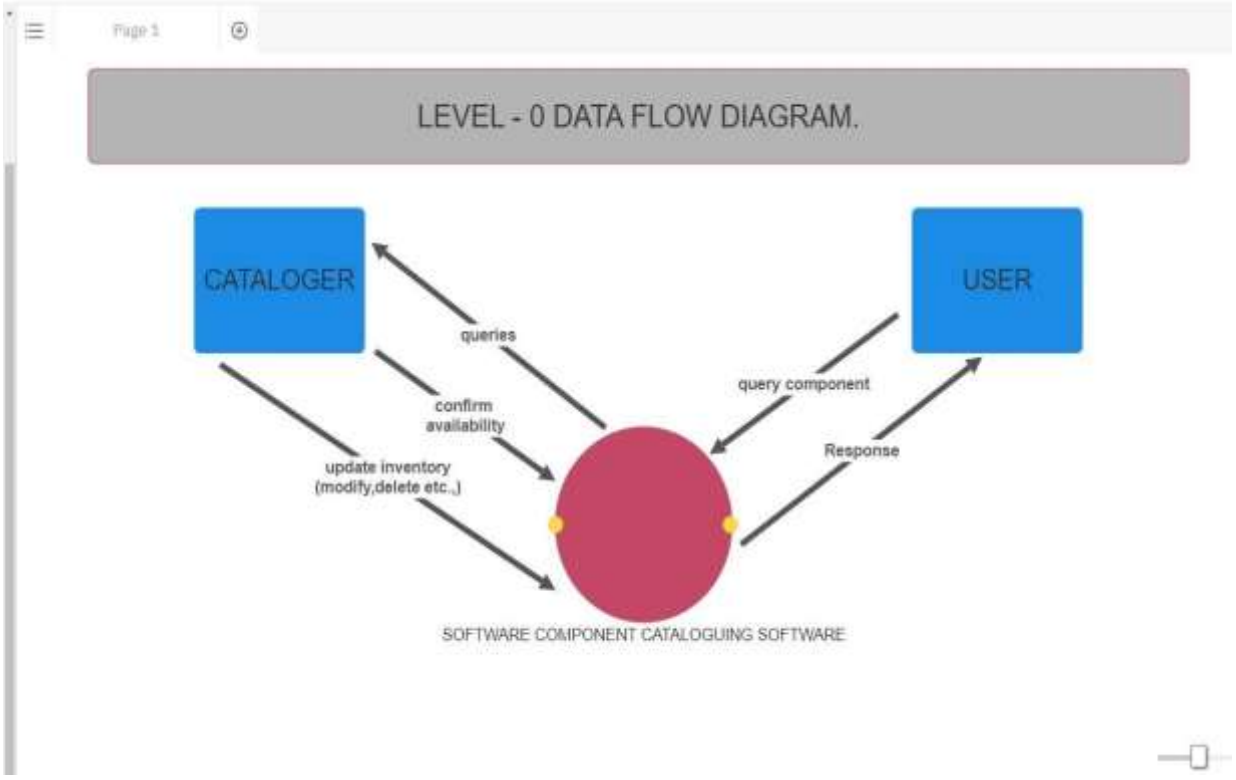
6.4. Sequence Diagram



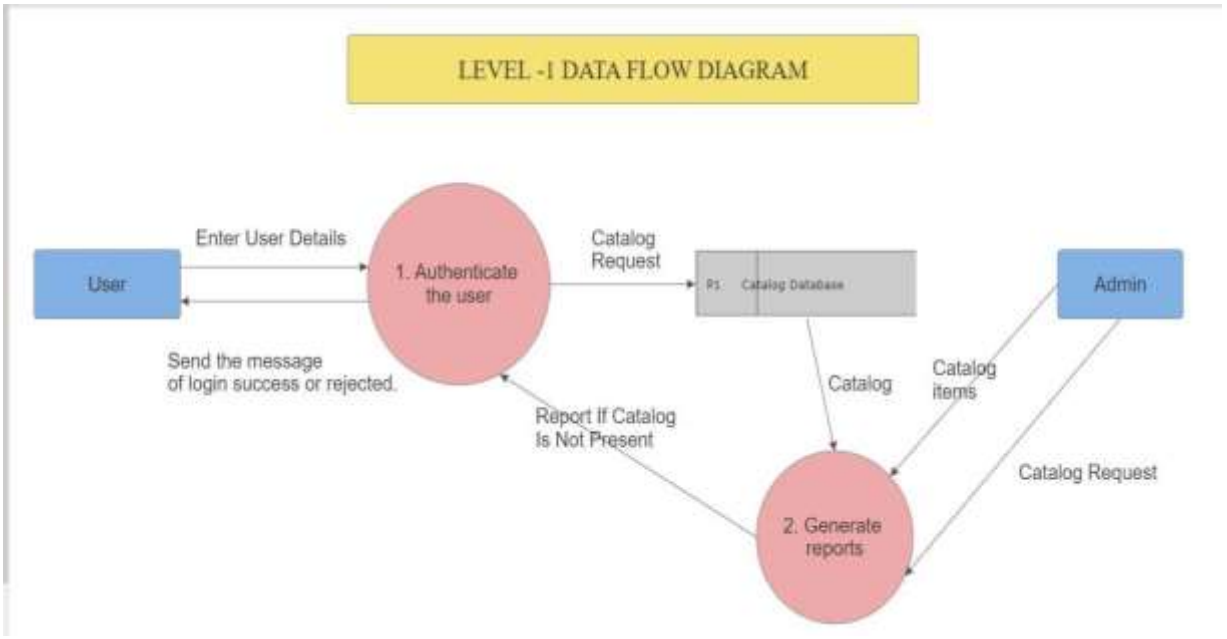
FOR GENERATING REPORTS TO THE DATABASE SERVER IN SCCS.

6.5. Data Flow Diagrams

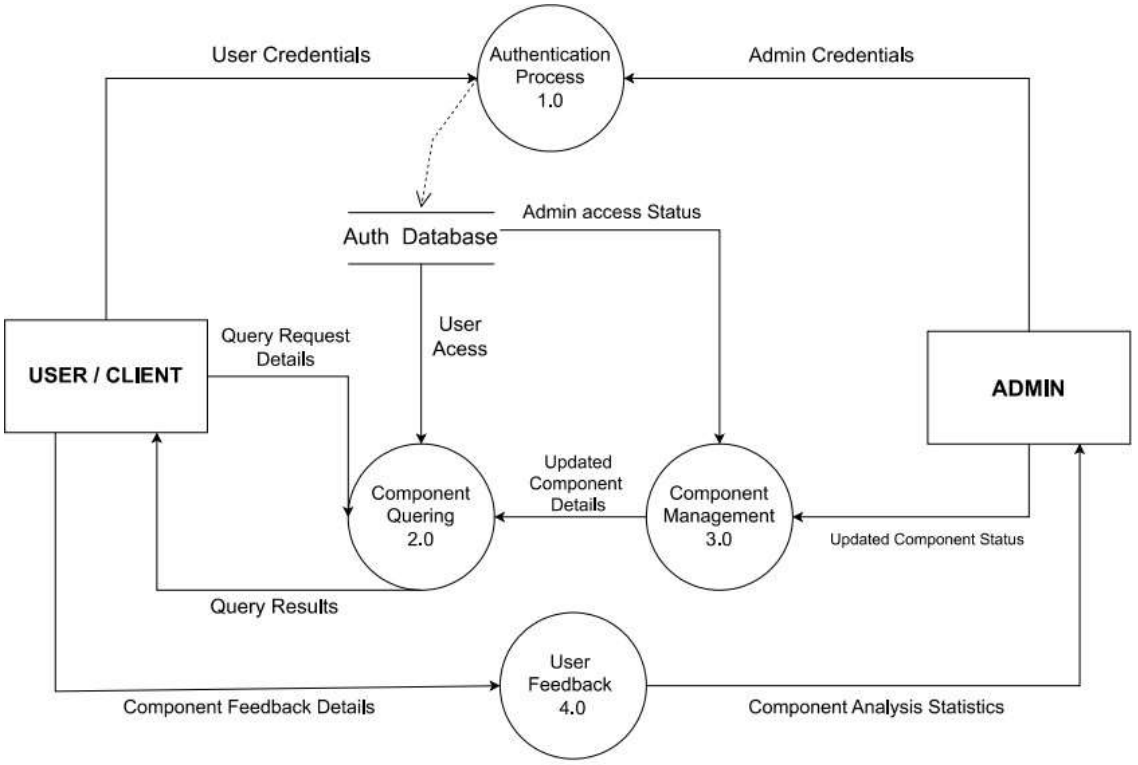
LEVEL-0 DATAFLOW DIAGRAM



LEVEL -1 DATAFLOW DIAGRAM (1)



LEVEL -1 DATAFLOW DIAGRAM (2)

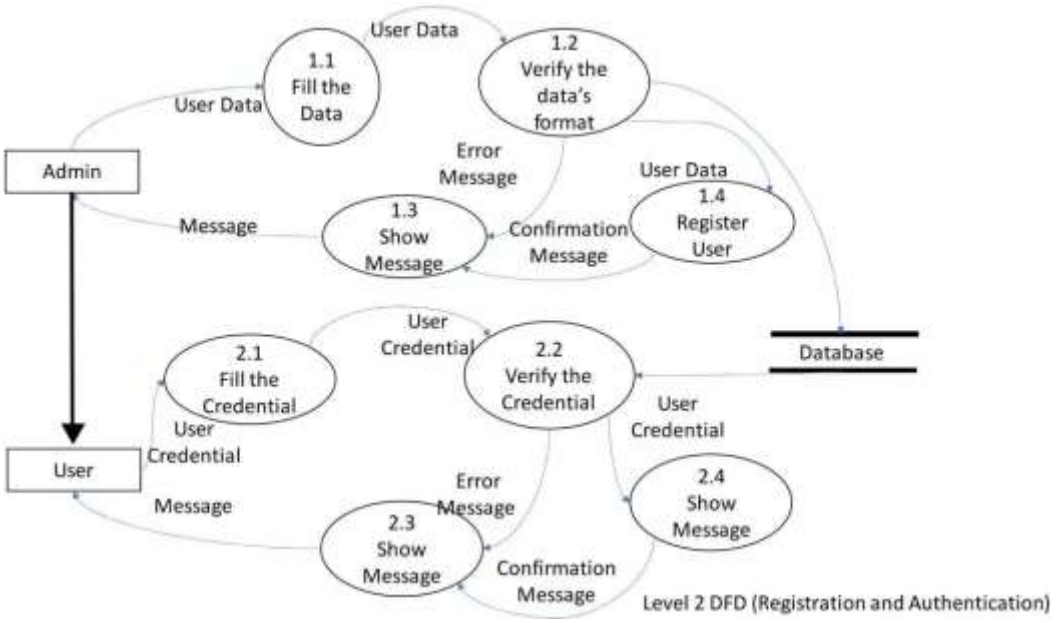


Level 1

Data Flow diagram

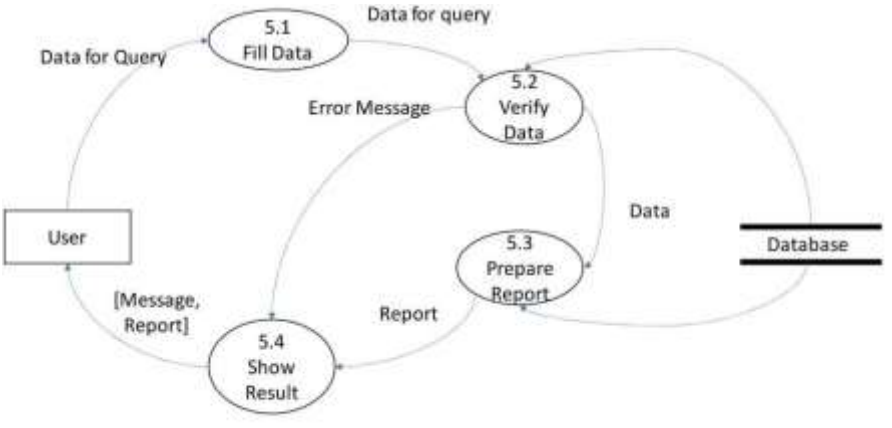
LEVEL – 2 DATAFLOW DIAGRAM

REGISTRATION AND AUTHENTICATION:



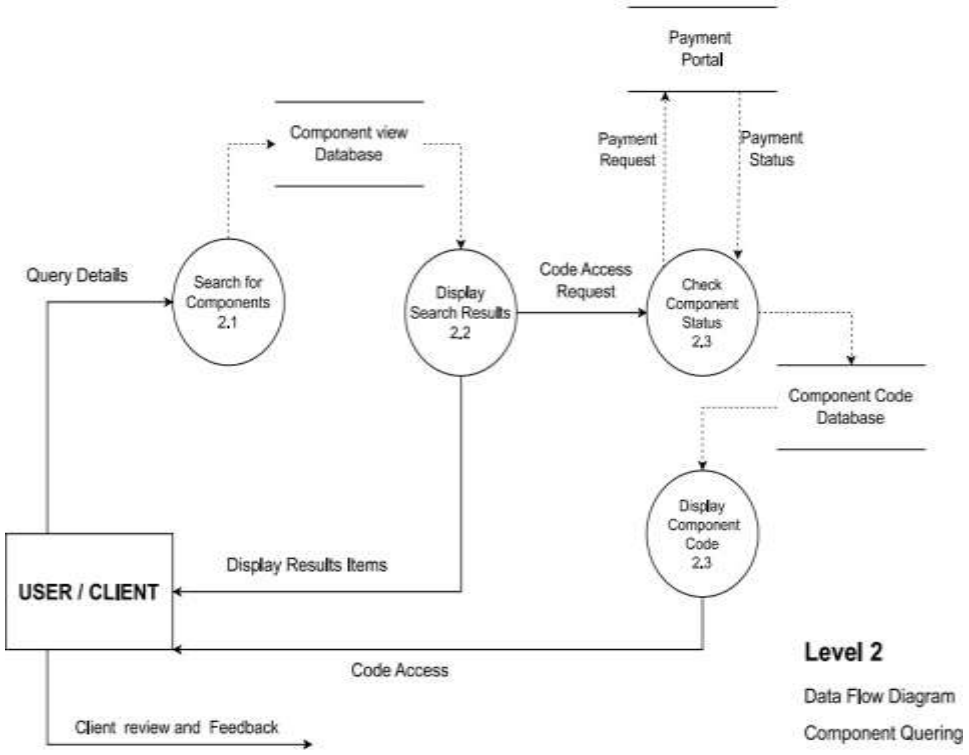
Level 2 DFD (Registration and Authentication)

REPORT GENERATION:

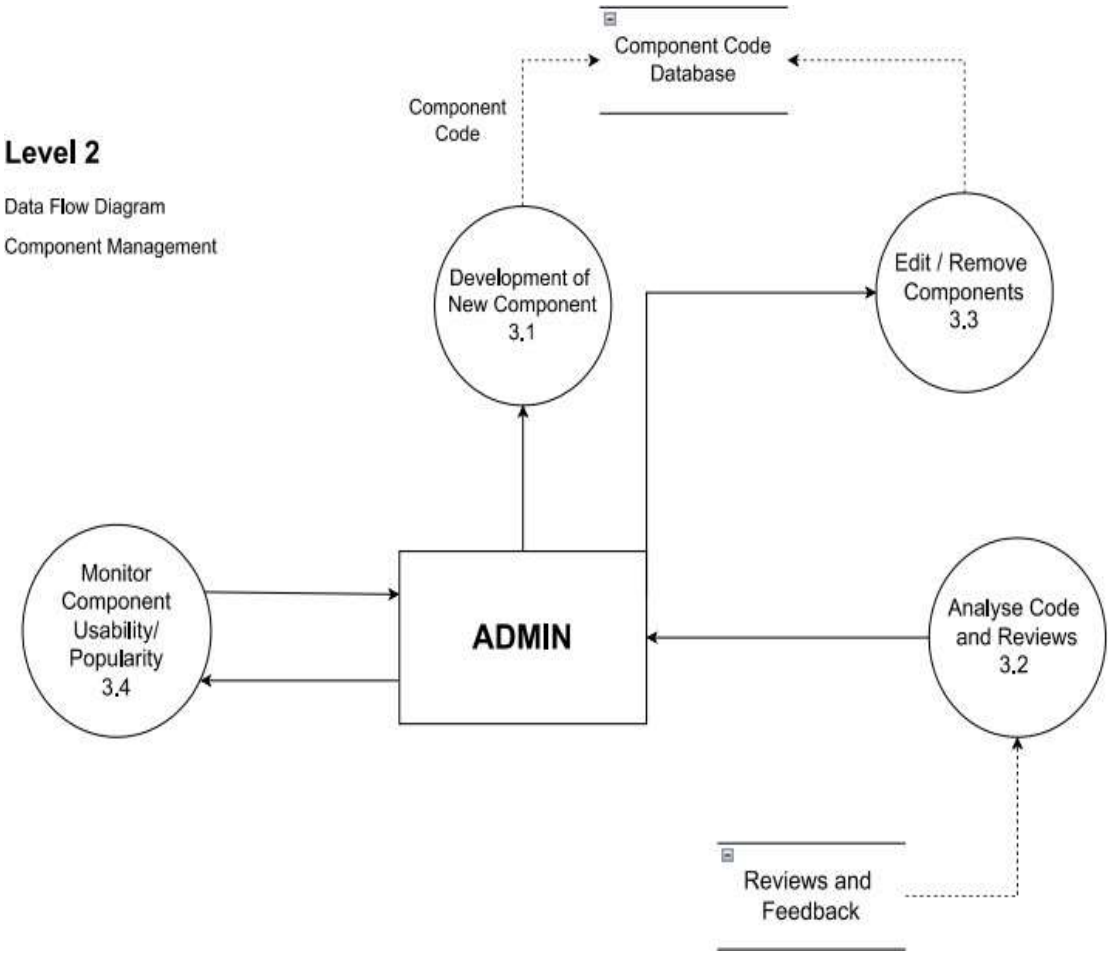


Level 2 DFD (Report Generation)

COMPONENT QUERYING:



COMPONENT MANAGEMENT:



7. IMPLEMENTATION DETAILS

USER INTERFACE:

FIGURE-1

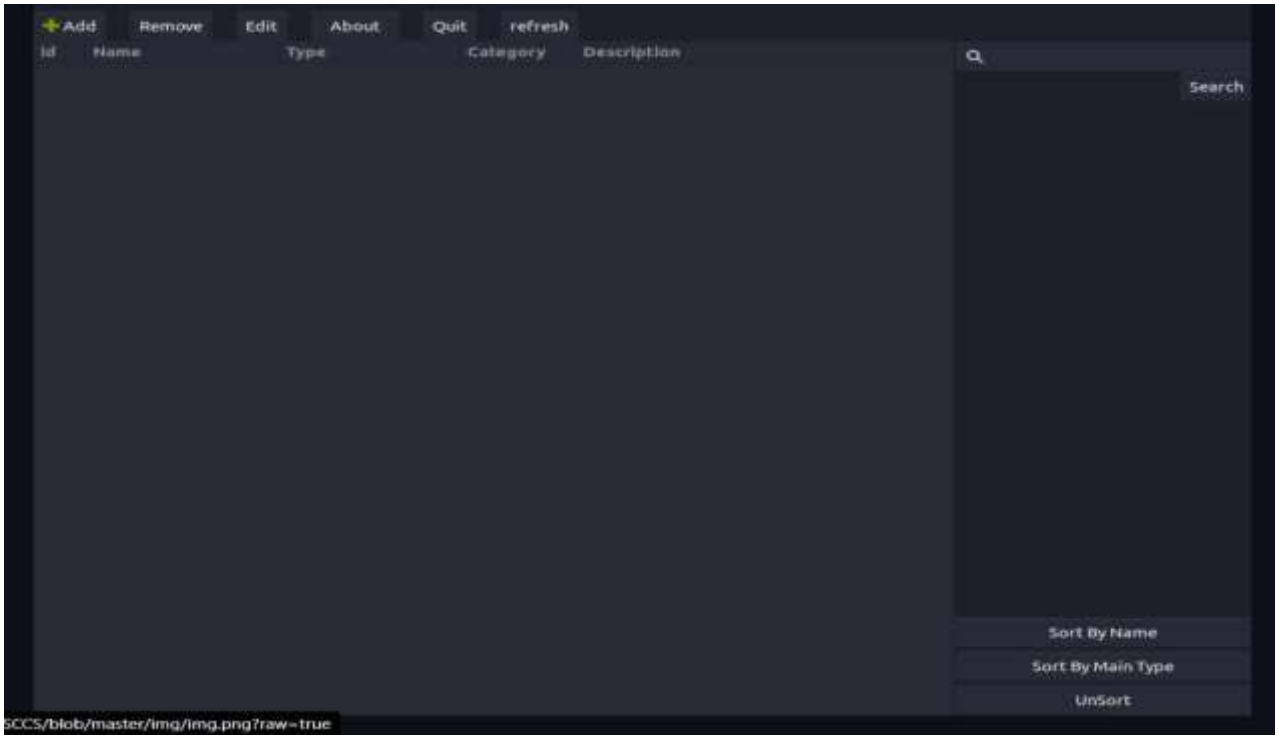


FIGURE-2

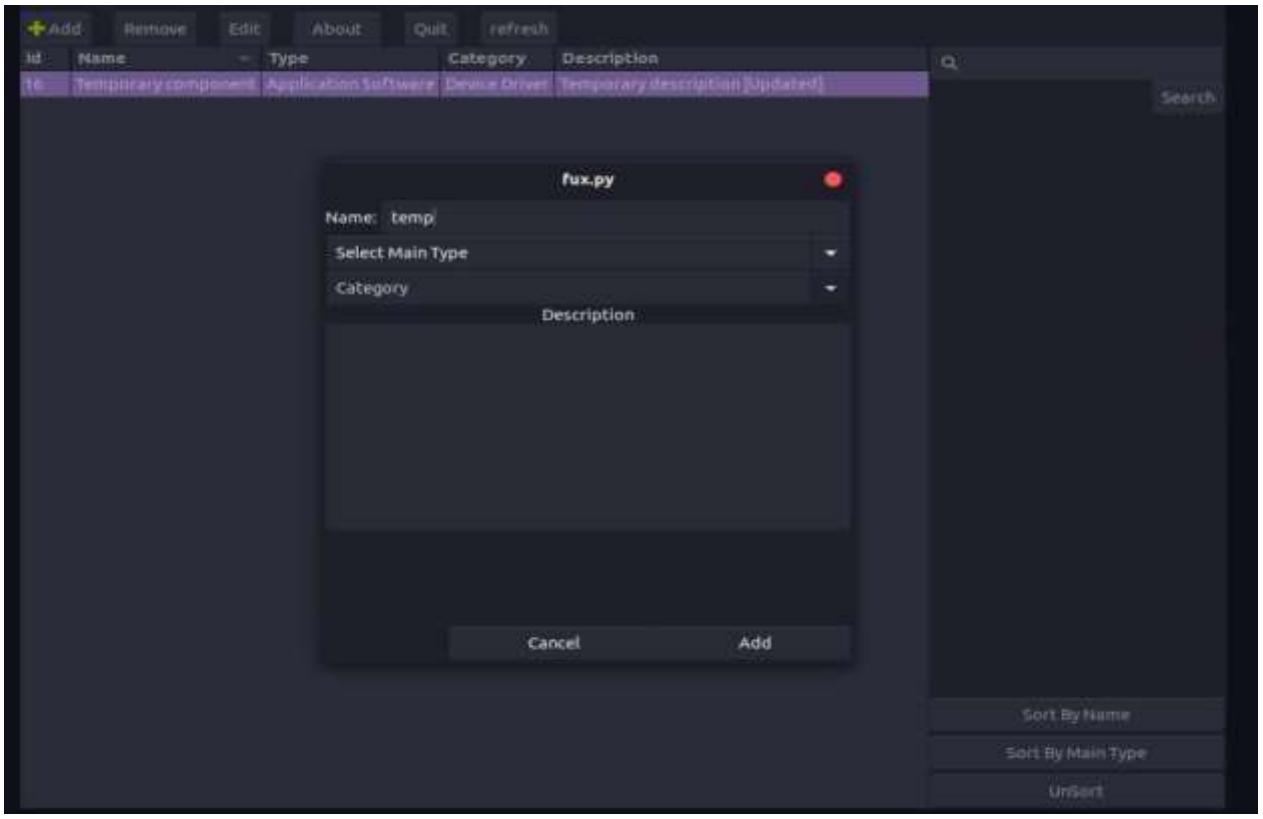


FIGURE-3

DATA ADDED:

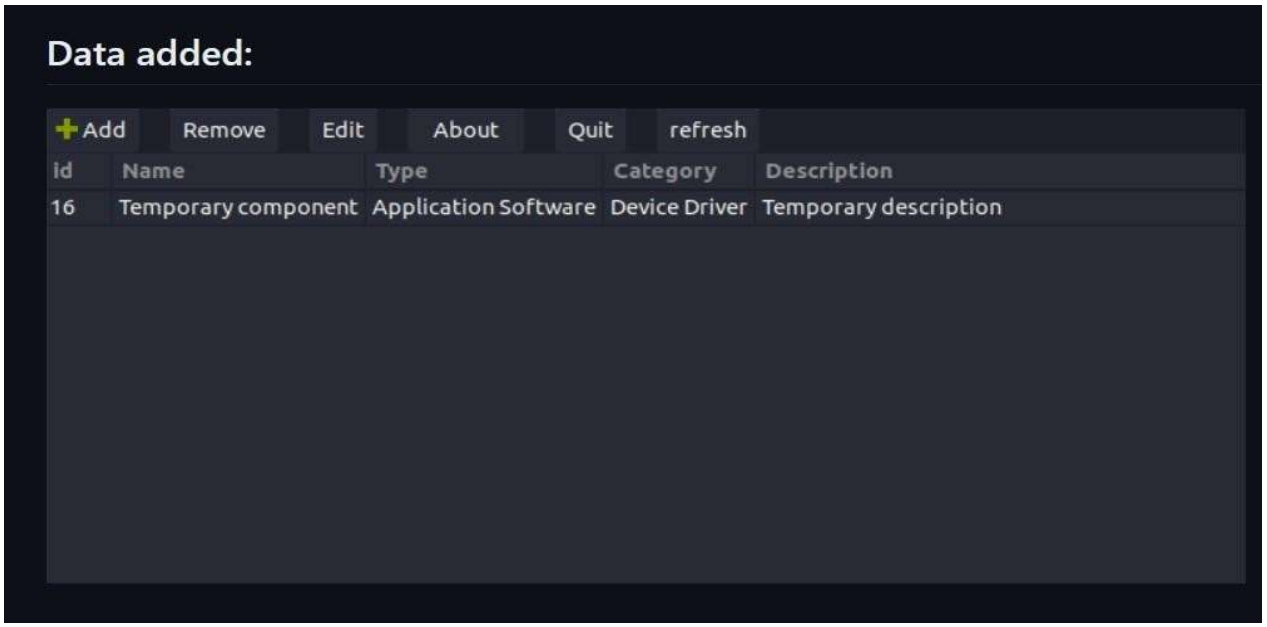
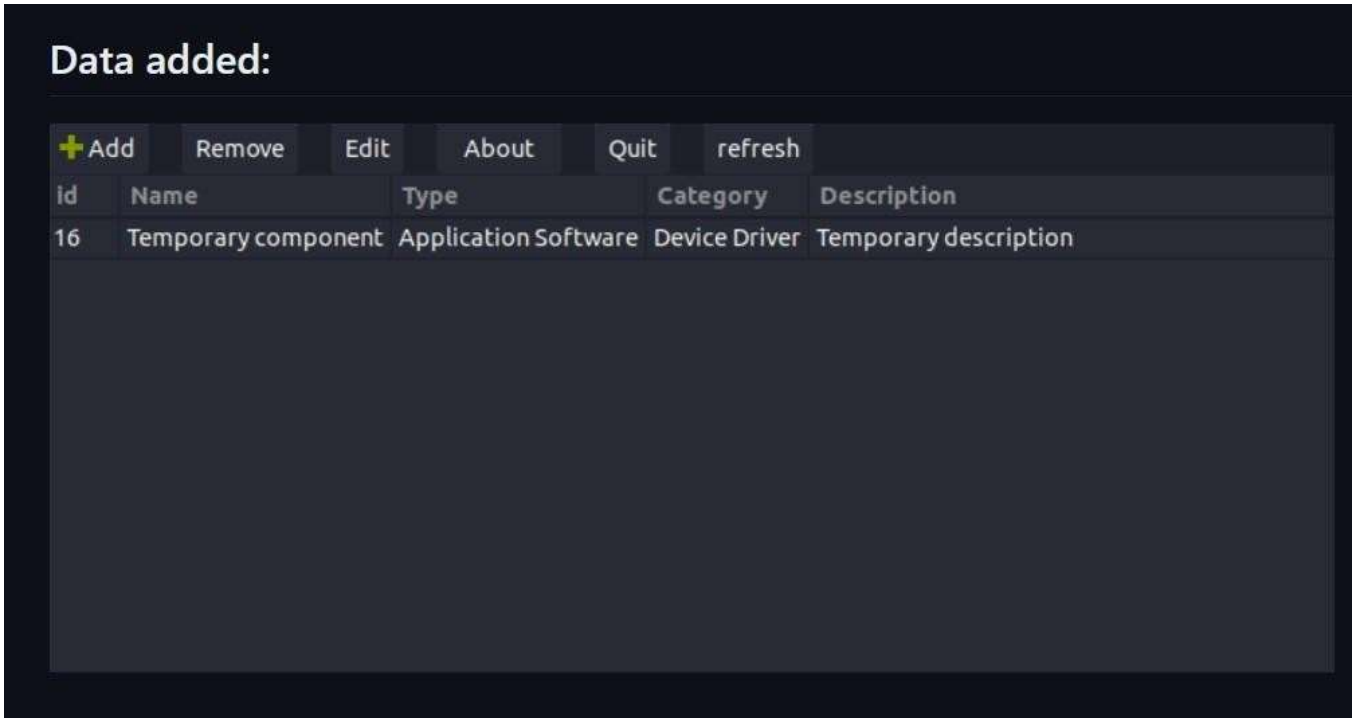


FIGURE-4

UPDATE DATA:



8. RESULTS AND DISCUSSIONS

The results obtained by using this Software component and catalog software are as follows:

1. **Improved Software Management:** The Software Component and System Software can significantly enhance software management processes by streamlining cataloging, version control, dependency management, and documentation inclusion, leading to more efficient software development cycles.

2. **Enhanced Collaboration:** The system facilitates better collaboration among development teams, stakeholders, and IT personnel by providing a centralized platform for accessing, evaluating, and sharing software components, configurations, and system architecture..

3. **Increased Productivity:** With reduced time spent on manual administrative tasks and improved software management practices, teams can focus more on core development activities, resulting in increased productivity, faster issue resolution, and smoother system integrations.

4. **Informed Decision-Making:** The system provides valuable analytics and insights that enable informed decision-making regarding software components, system configurations, performance optimizations, and overall system architecture.

Some remarks about drawback of the system are as follows:

1. **Implementation costs:** Implementing the Software Component and System Software may involve initial setup costs, including software acquisition, training, and potential infrastructure changes, which need to be carefully considered and budgeted for..

2. **User Adoption Challenges:** There may be resistance to change among users, particularly IT personnel, leading to slower adoption rates or lower utilization of the software, necessitating effective change management strategies, training programs, and user support.

3. **Technical Challenges:** Technical issues such as system compatibility, software dependencies, integration complexities, or performance bottlenecks can impact productivity and require prompt resolution and ongoing system maintenance to ensure

optimal system performance and reliability.

In conclusion, the Software Component and System Software can offer significant benefits to organizations by improving software management, collaboration, productivity, and decision-making processes. However, there may be potential drawbacks such as implementation costs, user adoption challenges, and technical complexities that need to be addressed and managed effectively to maximize the benefits of the software.

9. CONCLUSION

In conclusion, the proposed Software Component and Catalogue Software have the potential to significantly improve the efficiency of software management and development processes. The software can streamline cataloging, version control, dependency management, and documentation inclusion, providing a centralized platform for users to access, evaluate, and collaborate on software components effectively. However, there may be potential drawbacks to implementing the software, including initial setup costs, user adoption challenges, technical complexities, and data security considerations.

Overall, if the benefits of the Software Component and Catalogue Software outweigh the potential drawbacks, and the organization is committed to investing in the necessary resources for implementation, training, and maintenance, then the software can be a valuable tool for optimizing software management processes, enhancing collaboration, and accelerating software development cycles.

REFERENCES:

Here are the three main platforms which I used for software component cataloguing software:

1. **Maven Central:**
   - **Description: Maven Central is a repository for Java libraries and components used**
   - **inMaven-based projects.**
   - **Website: https://search.maven.org/**
2. **NuGet Gallery:**
   - **Description: NuGet Gallery is a repository for .NET libraries and components used**
   - **inNuGet-based projects.**
   - **Website: https://www.nuget.org/**
3. **npm Registry:**
   - **Description: npm Registry is a repository for JavaScript libraries and components**
   - **usedin Node.js and frontend projects.**
   - **Website: https://www.npmjs.com/**