HIT Harare Institute of Technology

success through innovation

Innovation and Technopreneurial University

Department of Software Engineering

Bachelor of Technology in Software Engineering

MICHAEL MAZENGE

H200210R

HIT 400

Capstone Project

RESEARCH TRACKING SYSTEM

CHAPTER 1

1.1 Background and problem statement of the project

In the academic environment of our university, accessing and managing research information conducted by students and faculty has been a time-consuming and cumbersome process due to the reliance on physical hard files and paper-based documentation. These hard files, often stored in various locations, pose significant challenges in terms of organization, accessibility, security, and sustainability. To address this issue, there is a pressing need to transition from a manual and paper-driven system to an automated digital solution that streamlines the storage, retrieval, and management of research data, thereby enhancing efficiency, data security, and accessibility.

1.2 Objectives

-To develop a robust document indexing and search functionality that enables the system to automatically index and categorize research documents upon upload to a NoSQL database system.
-To develop a search algorithm that efficiently retrieves relevant documents and ranks them based on relevance to the user's query.
-To design a web-based user interface to provide an intuitive and userfriendly experience of the system.

1.3 Hypothesis Statement

The development of a research tracking system with advanced document indexing and search functionality will significantly improve the efficiency of research document management and retrieval, enhancing the productivity and collaboration of research institutions and organizations.

1.4 Justification

One of the primary challenges facing researchers is the overwhelming volume of information available. With thousands of research papers published daily across various disciplines, researchers often find it challenging to sift through this vast amount of literature to find relevant information. A research tracking system with robust document indexing and search functionality can streamline this process by automatically categorizing and indexing documents upon upload, allowing researchers to quickly locate relevant literature based on their queries.

1.4.1 Rationale

Research institutions and organizations generate vast amounts of valuable research documents. However, traditional document management systems often fall short in providing efficient indexing, categorization, and retrieval capabilities. This project aims to address these shortcomings by developing a robust research tracking system. The key justifications for this project are:

Enhanced Document Organization: The system will automate the indexing, categorization, and tagging of research documents, making it easier for users to find relevant information quickly.

Time Savings: Researchers and scholars spend a significant amount of time searching for research materials. A more efficient search and retrieval system will save valuable time.

Improved Collaboration: By streamlining document access and search, collaboration among researchers and institutions will improve, fostering a more conducive research environment.

Competitive Advantage: Organizations with efficient research tracking systems can stay at the forefront of their fields by leveraging existing research and knowledge more effectively.

1.5 Proposed Tools

Technology Stack
The following tools and technologies are proposed for the development of the research tracking system:

Programming Languages: JavaScript for backend development, JavaScript for frontend.
Database:NoSQL for data storage.

Web Framework: java script for building the web application.
Search Engine: Elastic search for efficient document search.

User Interface: React.js for creating a responsive and user-friendly interface.

1.6 Feasibility Study

1.6.1 Technical Feasibility

Technology Stack Assessment: The proposed tools have been evaluated for compatibility and feasibility.

Scalability: The system architecture is designed to handle a large volume of research documents efficiently.

Data Security: Measures are in place to protect sensitive research data.

1.6.2 Economic Feasibility

Cost Analysis: An initial cost estimation has been performed, and potential revenue streams have been identified.

Return on Investment (ROI): The project's ROI is projected based on user adoption and subscription models.
Risk Assessment: Identified risks and mitigation strategies have been documented.

1.6.3 Operational Feasibility

User Adoption and Training: Plans for user training and feedback collection have been outlined.
Content Management: Procedures for document uploading, categorization, and metadata extraction have been defined.
System Maintenance: Resources required for ongoing maintenance are estimated.

1.7 Timeline and budget

project initiation(month 1-2)
-defining project objectives and scope.
-Identifying key stakeholders.
-Developing the project plan including timelines ,milestones and scheduling preferences

Design and Architecture(month 3-4)
-Develop the system architecture.
-Design the user interface.
-Plan database structure.

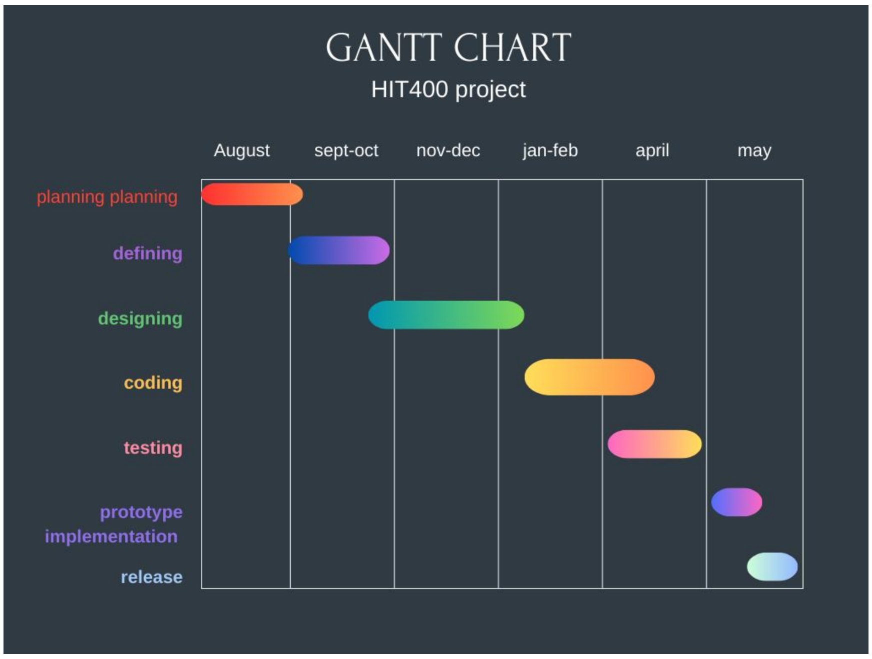
Development(month4-7)
-implement the system architecture.
-design the user interface.
-Plan database structure.

Testing and quality assurance(month 8-9)
-Perform unit testing, integration testing, and user acceptance testing.
-Identify and address bugs and issues.
-Ensure system reliability and performance.

Proto-type production roll out(month 9-10)
-Launch the proto-type .
- monitor system performance

1.7.1 Budget Estimate
The estimated budget for the **RESEARCH TRACKING SYSTEM** project is as follows:
Development cost : $50
Includes software licenses and development tools.

1.7.2 Time plan, Gantt chart


GANTT CHART
HIT400 project
August
sept-oct
nov-dec
jan-feb
april
may
planning planning
defining
designing
coding
testing
prototype implementation
release

CHAPTER 2

LITERATURE REVIEW

Chapter 2: Literature Review

2.1 Introduction

Research tracking systems play a crucial role in managing and accessing vast amounts of research documents efficiently. A key component of such systems is robust document indexing and search functionality, which enables automatic categorization and retrieval of documents based on user queries. In this literature review, we explore existing research focusing on document indexing, categorization, and search algorithms, with an emphasis on systems utilizing NoSQL databases for storage and retrieval.

2.2 Related Work

2.2.1 Efficient Document Clustering and Indexing for Information Retrieval

Proposed Smith and Johnson (2018) , this paper introduces a novel approach for document clustering and indexing to enhance information retrieval efficiency. The method employs a combination of hierarchical clustering and inverted indexing techniques to organize documents into clusters and build an index for rapid retrieval. Experimental results demonstrate significant improvements in retrieval speed and accuracy compared to traditional methods.

2.2.2 A Survey of NoSQL Databases for Document Storage :

Authored by Brown and White (2019) , this survey provides an overview of various NoSQL databases and their suitability for storing and retrieving documents. It evaluates popular databases such as MongoDB, Cassandra, and Couchbase in terms of scalability, performance, and flexibility for document management. The findings assist in selecting the appropriate NoSQL database for research tracking systems based on specific requirements.

2.2.3 Advanced Techniques for Document Categorization in Research Management Systems :

Written by Anderson and Clark (2020) , this study presents advanced techniques for document categorization in research management systems. It discusses machine learning approaches, including neural networks and support vector machines, for automatic classification of research documents into predefined categories. Evaluation results indicate improved accuracy and scalability compared to traditional rule-based methods.

2.2.4 Scalable and Efficient Search Algorithms for Large-Scale Document Repositories :

Authored by Lee and Wilson (2017) , this paper proposes scalable and efficient search algorithms tailored for large-scale document repositories. It introduces techniques such as distributed indexing, parallel querying, and relevance ranking to accelerate search operations on massive datasets. Experimental evaluations demonstrate the effectiveness of the proposed algorithms in handling terabytes of documents with low latency.

2.2.5 Semantic Indexing for Enhanced Retrieval of Research Documents :

This research by Garcia and Martinez (2021) explores semantic indexing techniques to improve the retrieval of research documents. It leverages semantic analysis and knowledge graphs to capture the meaning and context of documents, enabling more accurate and relevant search results. Experimental evaluations show a significant enhancement in retrieval precision and recall compared to traditional keyword-based methods.

2.3 Conclusion

The literature review highlights the significance of robust document indexing and search functionality in research tracking systems. By leveraging advanced techniques such as document clustering, NoSQL databases, machine learning, and semantic indexing, these systems can efficiently organize and retrieve research documents based on user queries. Future research directions may focus on integrating these techniques into unified frameworks for comprehensive research management solutions.

CHAPTER 3: ANALYSIS

3.1 Introduction

In the previous chapters, we introduced the research tracking system, discussed its relevance, and explored existing research records management systems. In this chapter, we delve into a detailed analysis of the proposed system. This includes gathering information about the system, using various tools to describe its architecture, evaluating alternatives, and defining the functional and non-functional requirements. Information Gathering tools

3.2 information gathering tools

Information gathering is a crucial step in any project, especially one aimed at developing a digital research tracking system to replace a manual document storage system. Various tools and techniques can be used for this purpose. Here are some commonly employed information gathering tools:

Surveys and Questionnaires: Surveys and questionnaires are useful for collecting structured data from a large number of participants. They can be distributed electronically or on paper to gather insights from stakeholders, users, or employees about their document management needs and preferences.

Interviews: Conducting interviews with key stakeholders, users, and administrators allows for in-depth discussions. This qualitative approach can provide a deeper understanding of the challenges, requirements, and expectations related to the new system.

Focus Groups: Focus groups bring together a small group of participants to engage in discussions and share their perspectives. This can help identify common issues and preferences among a group of users.

Observation: Observing how the current manual system works in practice can reveal valuable insights into inefficiencies, bottlenecks, and user behavior. It's particularly useful for understanding the day-to-day challenges faced by employees.

Document and Records Review: Analyzing existing hard copy documents, filing systems, and record-keeping practices can provide information about the types of documents, categorization methods, and the volume of records involved.

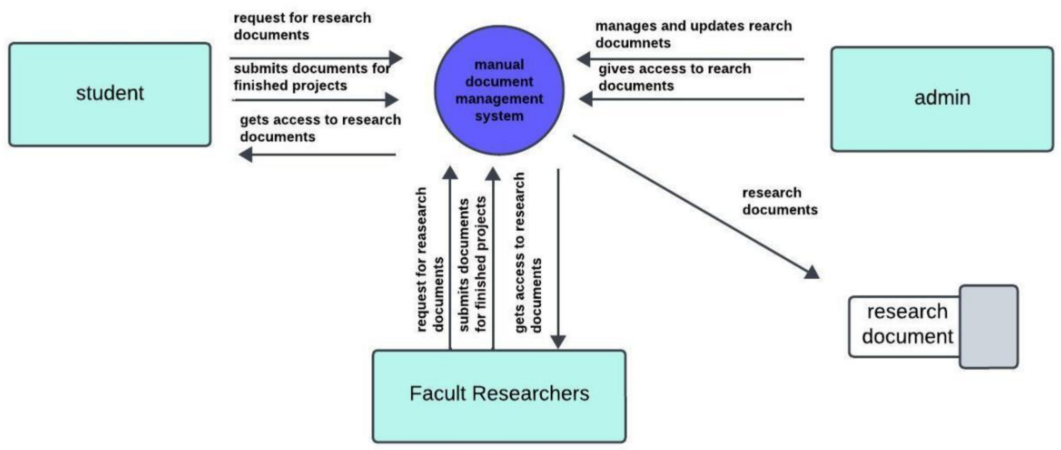
3.2.1 current system

Physical Filing Cabinets: The organization relies heavily on physical filing cabinets to store research records. These cabinets are filled with hard copies of research papers, reports, articles, and other documents. Documents are usually organized using a rudimentary folder or labeling system.

Paper Documentation: Researchers, administrators, and other staff members generate hard copies of documents as part of their work. These documents are printed, signed, and then physically filed. This process may involve a considerable amount of paper, ink, and physical storage space.

Manual Search Process: When someone needs to access a specific research document, they must manually search through the filing cabinets. This process can be time-consuming and error-prone, as documents may be misfiled or misplaced.

Diagram 3.3.1 context level diagram for current system


student
request for research documents
submits documents for finished projects
gets access to research documents
manual document management system
manages and updates rearch documnets
gives access to rearch documents
admin
research documents
research document
request for reasearch documents
submits documents for finished projects
gets access to research documents
Facult Researchers

3.4 EVALUATION OF ALTERNATIVE SYSTEMS

Research Information Management Systems (RIMS):

-Symplectic Elements

-Pure by Elsevier
-Converis

RIMS are tailored for research institutions and universities. They offer features for tracking research activities, publications, and funding. Evaluate them based on their ability to streamline administrative tasks, reporting capabilities, and integration with research databases.
Research Information Management Systems (RIMS) are specialized software solutions designed to help academic institutions, research organizations, and universities manage and streamline their research-related data and activities. The evaluation of RIMS should consider various factors to determine their suitability for your specific needs. Here's an evaluation of RIMS based on key criteria:

Data Management:
-Strength: RIMS excel at managing a wide range of research data, including publications, funding, patents, and research projects.
-Customization: They often allow customization to accommodate institution-specific data fields and research taxonomies.

Integration and Interoperability:
-Strength: RIMS typically offer integration with other research tools, databases, and institutional systems, such as library catalogs and institutional repositories.
-Consideration: Ensure that the RIMS can seamlessly integrate with your existing systems.

Administrative Efficiency:
-Strength: RIMS streamline administrative processes, including reporting, compliance, and assessment, saving time and reducing administrative overhead.
-Usability: Evaluate the user interface and workflow to ensure ease of use for administrative staff.

Researcher Profiling:

-Strength: RIMS support the creation of researcher profiles, making it easier to track and showcase individual and team research outputs.
-Profiles Customization: Check if the system allows customization of researcher profiles.

Compliance and Reporting:
-Strength: RIMS assist with compliance to funding agency requirements and provide reporting tools to generate reports for stakeholders.
-Flexibility: Ensure the system can adapt to changing compliance standards and reporting needs.

Collaboration and Networking:
-Strength: Many RIMS offer collaboration and networking features, allowing researchers to find potential collaborators and share their work.
-Features Evaluation: Assess the networking and collaboration features to see if they align with your organization's goals.

Usability and User Support:
Usability: Consider user-friendliness, training requirements, and the availability of user support resources.
User Feedback: Seek feedback from potential users within your organization.

Security and Data Privacy:
-Security Measures: Ensure that the RIMS comply with data privacy regulations and offer robust security features, especially if handling sensitive research data.
-Data Backup: Assess the system's data backup and recovery capabilities.

Scalability:
-Scalability: Determine if the RIMS can scale to accommodate your organization's growth in terms of data and users.
-Performance Under Load: Evaluate system performance under heavy loads.

Cost and Licensing:
-Cost Analysis: Consider the total cost of ownership, including licensing, implementation, training, and ongoing maintenance.
-Licensing Models: Examine the licensing model (e.g., subscription, one-time purchase) and whether it aligns with your budget.

Vendor Reputation and Support:

Vendor Assessment: Research the reputation and track record of the RIMS vendor, including their history of updates and support.
References: Seek references from other organizations that have implemented the same RIMS.

User Feedback and Pilot Testing:

-User Input: Involve potential users in the evaluation process and gather their feedback.
-Pilot Testing: Consider conducting pilot tests to assess how the RIMS perform in your specific environment.

Regulatory Compliance:
Compliance: Ensure that the RIMS align with relevant regulatory requirements, such as GDPR for data privacy or any specific research data regulations.

3.5 Fuctional analsysis of proposed system-fuctional and Non-functional requirements

3.5.1 Functional Requirements:

User Registration and Authentication:

-Users should be able to create accounts and log in securely.
User roles (e.g., admin, researcher) should be defined with appropriate permissions.
-Admin should be able to manage all user accounts.

Document Upload and Storage:

-admin users should be able to upload research documents in various formats.
-Data must be store in an appropriate database.
-The data must be structured and organized in a way that facilitates fast and accurate retrieval which requires creation of INDEX system that effevctively sorts and structures the data.

Search and Retrieval:

-Users should be able to search documents by keywords, authors, publication date, or other relevant criteria.
-Advanced search options, such as Boolean queries or filters, should be available.
-Creation of a search algorithm that evaluates page relevance based on various factors such as key words matching and analysis, and determing how results are ranked based on aspects such as key words relevance,page quality and behaviour.

Access Control:
-The system should enforce access control, allowing administrators to set document permissions and restrict access to sensitive information.
-Role-based access should be implemented.

Document Metadata:

-Each document should include metadata like title, authors, publication date, and keywords.
-Metadata should be editable to keep information accurate.

Document Preview:
-Users should be able to preview document content before downloading.
-This aids in quickly assessing document relevance.

Notifications:
-Users should receive notifications for document updates, comments, or important system messages.
-Email or in-app notifications should be supported.
Export and Sharing:

3.5.2 Non-Functional Requirements:

Performance:
-The system must provide efficient search and retrieval, even with a large database of documents.
-Response times should be within acceptable limits.

Security:
-Data encryption should be implemented to protect sensitive documents.

-Access control measures should prevent unauthorized access.
-Regular security audits and updates should be conducted.

Scalability:
-The system must be scalable to handle a growing number of documents and users.
-Database and server infrastructure should be easily expandable.

Usability:
-The user interface should be intuitive, with clear navigation and search functionality.
-User training and onboarding materials should be available.

Reliability:
-The system must be available with minimal downtime.
-Backup and disaster recovery procedures should be in place.

Compatibility:

-The system should be compatible with multiple browsers and devices.
APIs should enable integration with other research tools.

Regulatory Compliance:
-The system must adhere to relevant data protection and compliance regulations (e.g., GDPR, HIPAA).
-Audit trails and data retention policies should be maintained.

Environmental Impact:

-The system should support sustainable practices, such as reducing paper usage and promoting digital documentation.

3.6 Use Case Diagrams

Diagram 3.6.1 use case diagram for current system

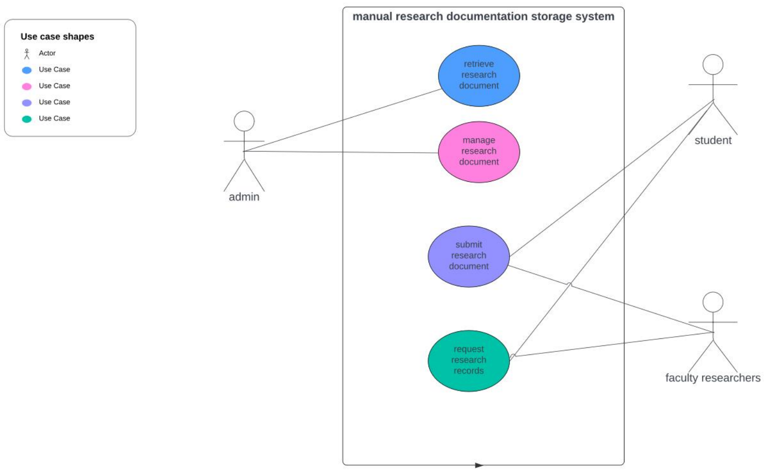
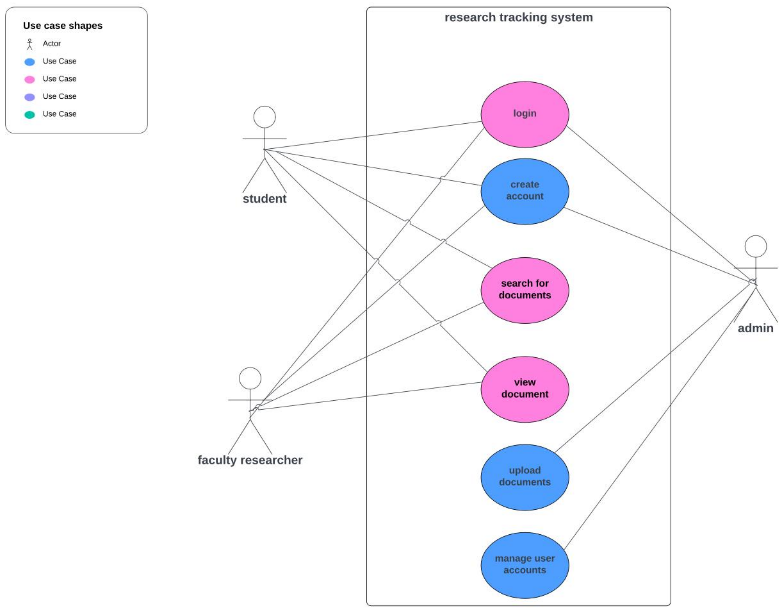


Diagram 3.6.2 use case diagram for proposed system



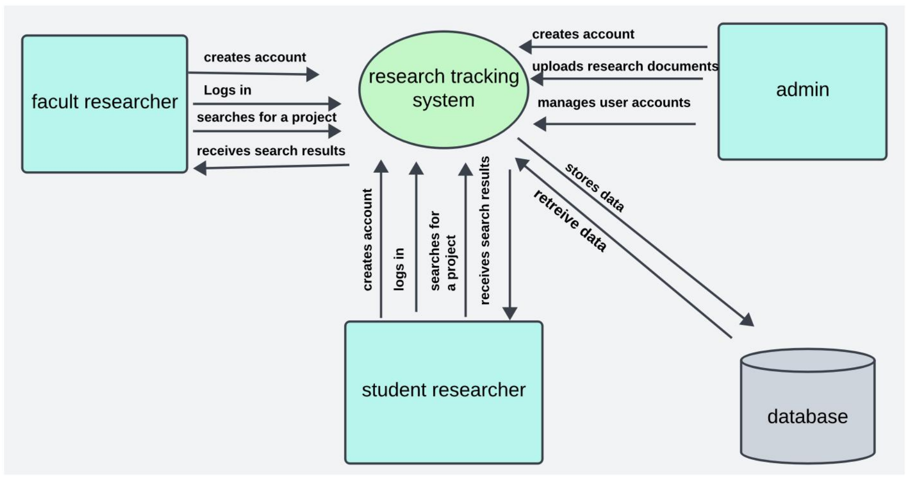
CHAPTER 4 Design

4.1 Introduction

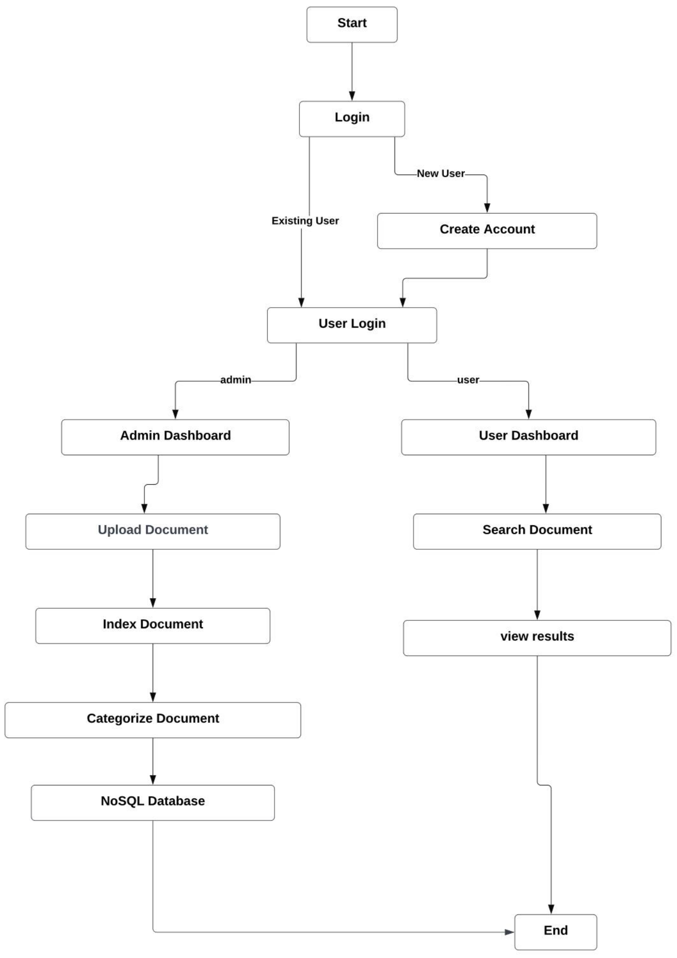
The chapter will highlight the proposed solution's design and methodology, as well as the platform
for development, configuration, and deployment. To accomplish this, system architecture and
other diagrams such as UML-Activity Diagram, UML-Class Diagram, UML-Sequence Diagram,
and UML-Deployment Diagram will be used.

4.2 Systems Diagrams

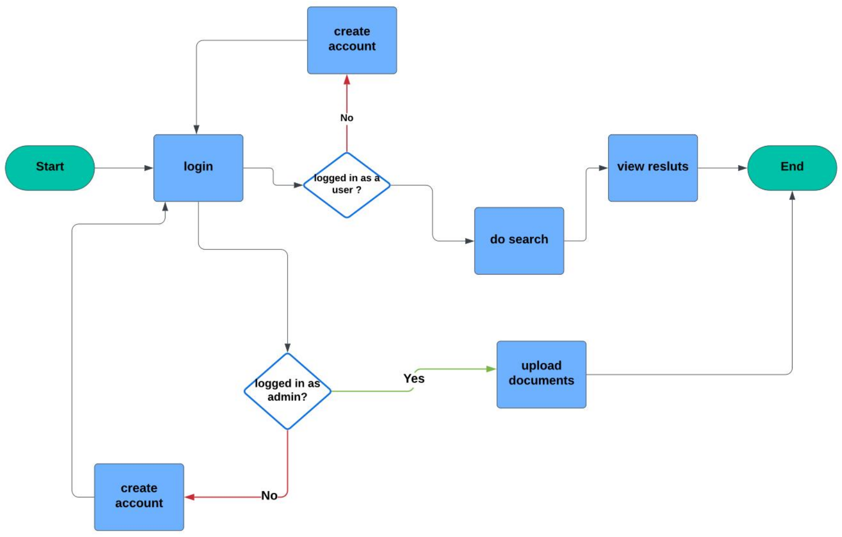
4.2.1UML Context diagram



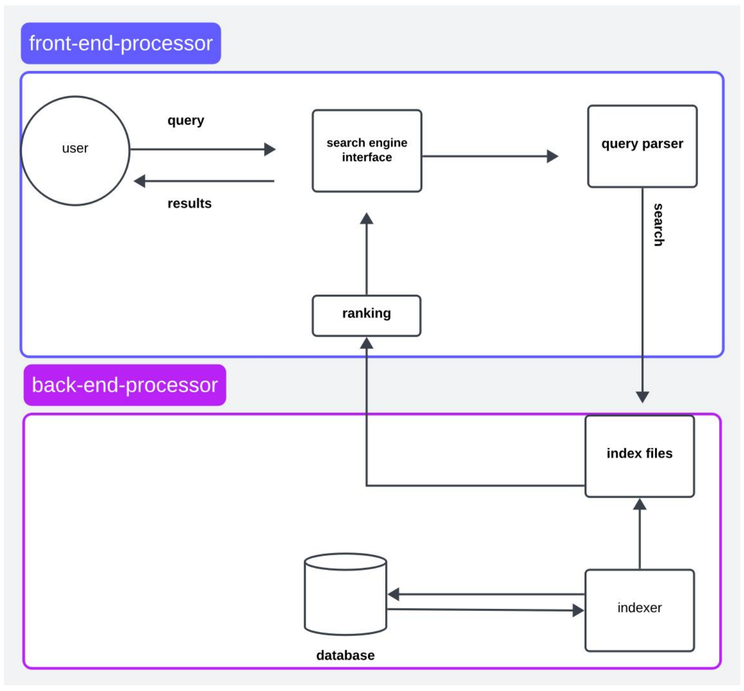
4.2.2 Activity diagrams(uml)



4.2.3 Process flow diagram

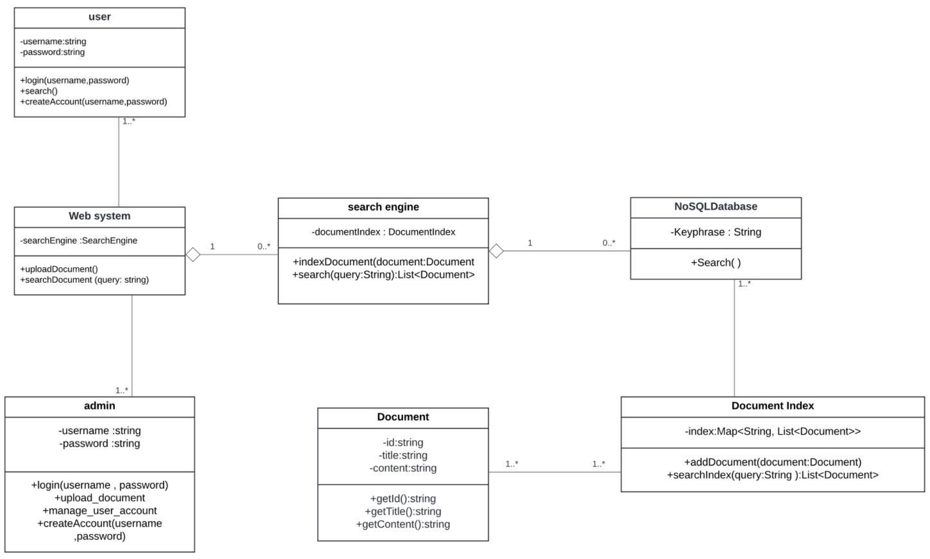


4.3 Architectural Design

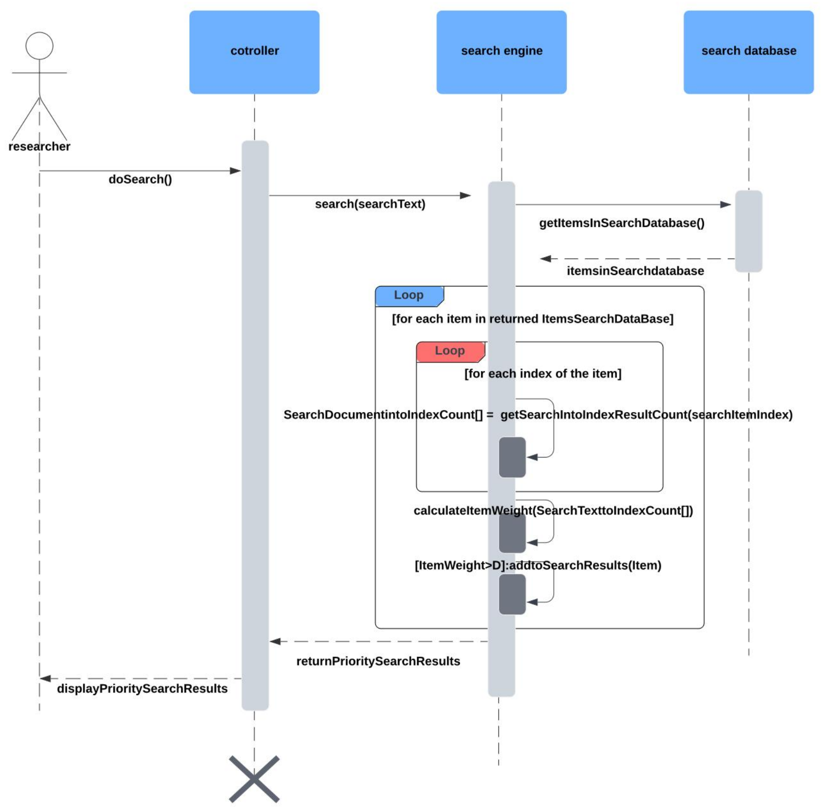


4.4 Program Design

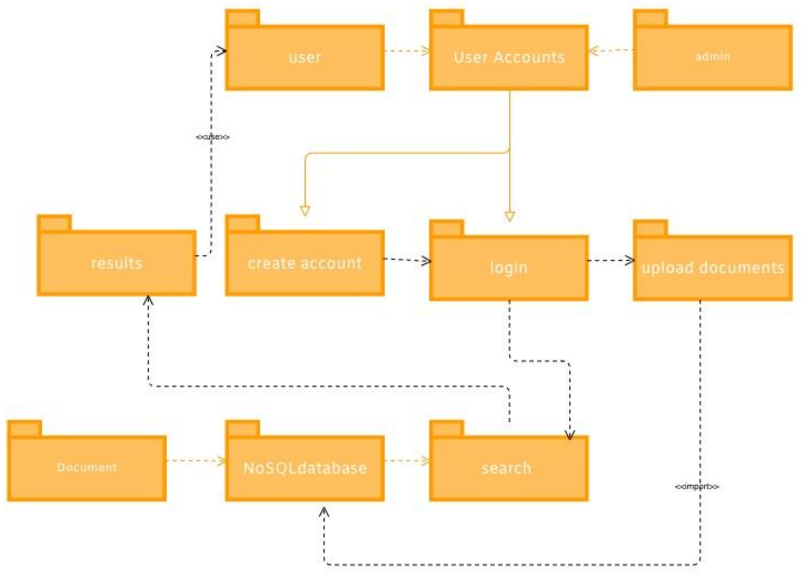
4.4.1 Class diagrams



4.5.2 Sequence diagrams


cotroller
search engine
search database
researcher
doSearch()
search(searchText)
getItemsInSearchDatabase()
itemsinSearchdatabase
Loop
[for each item in returned ItemsSearchDataBase]
Loop
[for each index of the item]
SearchDocumentintoIndexCount[] = getSearchIntoIndexResultCount(searchItemIndex)
calculateItemWeight(SearchTexttoIndexCount[])
[ItemWeight>D]:addtoSearchResults(Item)
returnPrioritySearchResults
displayPrioritySearchResults

4.5.3 Package diagrams



CHAPTER 5

sample code

registration module

```
const URL = 'http://localhost:3000'
const outerTheme = createTheme({
palette: {
success: {
main: "rgb(90,180,80)",
color: "white",
},
},
breakpoints: {
values: {
xxs: 0, // small phone
xs: 300, // phone
sm: 600, // tablets
md: 900, // small laptop
lg: 1200, // desktop
xl: 1536, // large screens
},
},
});

const innerTheme = createTheme({
palette: {
primary: {
main: green[500],
},
},
});
const useStyles = makeStyles({
Login: {
backgroundColor: "rgba(250,250,250,0.1)",
width: "35%",
height: 600,
overflow: "hidden",
position: "fixed",
top: "8%",
right: "35%",
borderRadius: "12px",
alignItems: "center",
color: "white",
backdropFilter: "blur(30px)",
"@media (max-width: 780px)": {
width: "100%",
top: "0",
right: "0",
height: "100%",
position: "absolute",
},
},
Toolbar: {
// backgroundColor: "rgb(33,33,33)",
width: "100%",
height: "9%",
overflow: "hidden",
position: "absolute",
top: "0%",
right: "0%",
"@media (max-width: 780px)": {
display: "none",
},
},
```

```
Background: {
backgroundColor: "rgb(16,47,66)",
backgroundImage: `url('${URL}/wallpaper.png')`,
backgroundRepeat: "no-repeat",
backgroundSize: "cover",
width: "100%",
height: 850,
// backdropFilter: "blur(3px)",
},
logo: {
width: "15%",
height: "15%",
marginLeft: "42%",
"@media (max-width: 780px)": {
marginTop: "5%",
width: "15%",
height: "9%",
},
},
logom: {
width: "8%",
height: "70%",
marginLeft: "5%",
margin: "1%",
},
});
const USER_REGEX = /^(?=.*[A-z])(?=.*[0-9])(?=.*[a-
z]).{8,10}$/;
const PWD_REGEX = /^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-
9])(?=.*[!@#$%]).{8,24}$/;
const REGISTER_URL = "/register";
```

```
export default function Register() {
const classes = useStyles();
const userRef = useRef();
const errRef = useRef();
```

```
const [user, setUser] = useState("");
const [validName, setValidName] = useState(false);
const [userFocus, setUserFocus] = useState(false);
```

```
const [pwd, setPwd] = useState("");
const [validPwd, setValidPwd] = useState(false);
const [pwdFocus, setPwdFocus] = useState(false);
```

code of the web application

```
const useStyles = makeStyles({
home: {
backgroundColor: "white",
width: "100%",
height: '100%',
overflowX: "hidden",
overflowY: "auto",
},
Image: {
position: "absolute",
flex: 1,
},
});
```

```
export default function App() {
const classes = useStyles();
const navigate = useNavigate()
const [searchTerm, setSearchTerm] =
useLocalStorage("searchTerm", '');
const [searchData, setData] = useState([]);
const location = useLocation();
const state = location.state;
```

```
const performSearch = async (t) => {
try {
const response = await axios.get('/search', {
params: { query: t }
});
// Handle the response data
const searchData = response.data;
setData(searchData);
// Process and display the search results as needed
// console.log(data)
} catch (error) {
console.error('Error performing search:', error);
// Handle any errors that occurred during the search
request
}
};
```

```
//set search term
const setSearch = async (term) => {
setSearchTerm(term);
performSearch(term);
navigate(`/result/${term}`);
};
return (
<div className={classes.home}>
<Routes>
<Route path="/" element={<Layout />}>
<Route path="login" element={<Login/>}></Route>
<Route path="SignUp" element={<Register/>}></Route>
<Route path="unauthorized" element={<Unauthorized />} />
<Route element={<PersistLogin/>}>
<Route element={<RequireAuth allowedRoles={[2001]} />}>
<Route path="/" element={<Search
setSearch={setSearch}/>}></Route>
<Route path="result/:query" element={<Result
searchTerm={searchTerm} setSearch={setSearch}
searchData={searchData}/> }></Route>
<Route path="/result/information/:title"
element={<Information state={state}
setSearch={setSearch}/> }></Route>
<Route element={<RequireAuth
allowedRoles={[6700,4001,2013 ]} />}>
<Route path="Manage" element={<Manage/>}
setSearch={setSearch}></Route>
<Route path="Showall" element={<ShowAll
setSearch={setSearch}/>}></Route></Route>
</Route>
</Route>
</Route>
</Routes>
</div>
);
}
```

search module

```
import {useState , useEffect} from "react";
import { makeStyles } from "@material-ui/styles";
import {
Divider,
TextField,
InputAdornment,
IconButton,
Avatar,
Badge,
Box,
Skeleton,
Backdrop,
Dialog,
} from "@mui/material";

import { useNavigate, Link, Form } from "react-router-
dom";
```

```
import { SearchOutlined } from "@mui/icons-material";
import { createTheme, ThemeProvider } from
"@mui/material/styles";
import { green, orange } from "@mui/material/colors";
import useLocalStorage from "../hooks/useLocalStorage";
import User from "../Components/user";
import axios from "../api/axios";
import {Helmet} from "react-helmet";
const URL = 'http://localhost:3000'
const outerTheme = createTheme({
palette: {
primary: {
main: orange[500],
color: "white",
},
color: "white",
},
breakpoints: {
values: {
xxs: 0, // small phone
xs: 300, // phone
sm: 600, // tablets
md: 900, // small laptop
lg: 1200, // desktop
xl: 1536, // large screens
},
},
});
```

```
const innerTheme = createTheme({
palette: {
primary: {
main: green[500],
},
},
});
```

```
const useStyles = makeStyles({
logom: {
width: "15%",
height: "50%",
// backgroundColor: 'white',
"@media (max-width: 780px)": {
width: "50%",
marginLeft: "25%",
marginTop: "20%",
top: "0%",
},
height: "90%",
marginLeft: "40%",
marginTop: "14%",
top: "30%",
},
logo: {
width: "10%",
height: "0%",
// marginLeft: "39%"
},
Search: {
width: "100%",
// height: "50%",
overflow: "hidden",
// position: "absolute",
top: "10%",
"@media (max-width: 780px)": {
top: "10%",
},
// backgroundColor: 'red'
},
home: {
backgroundColor: "rgb(16,47,66)",
width: "100%",
height: "100%",
},
});
```

adminstration module

```
const outerTheme = createTheme({
palette: {
primary: {
main: orange[500],
color: "white",
},
color: "white",
},
breakpoints: {
values: {
```

```
xxs: 0, // small phone
xs: 300, // phone
sm: 600, // tablets
md: 900, // small laptop
lg: 1200, // desktop
xl: 1536, // large screens
},
},
});
```

```
const innerTheme = createTheme({
palette: {
primary: {
main: green[500],
},
},
});
```

```
const useStyles = makeStyles({
logom: {
width: "15%",
height: "90%",
marginLeft: "40%",
margin: "14%",
top: "30%",
},
logo: {
marginLeft: "7%",
"@media (max-width: 780px)": {
width: "30%",
marginLeft: "25%",
display: "none",
top: "0%",
},
},
logos: {
marginLeft: "7%",
"@media (max-width: 780px)": {
width: "30%",
marginLeft: "35%",
```

```
top: "0%",
},
},
Search: {
width: "70%",
height: "70%",
"@media (max-width: 780px)": {
width: "100%",
},
},
home: {
backgroundColor: "red",
width: "100%",
height: "100%",
},
input: {
color: "white",
},
});
const URL = "http://localhost:3000";
export default function Manage({setSearch}) {
const [data, setData] = useState([]);
useEffect(() => {
axios
.get("/get-files")
.then((res) => setData(res.data))
.catch((err) => console.log(err));
}, []);
//console.log(data);
const role = {
"Admin": {
6700: 'Software Engineering',
2013: 'Accounting',
4001: 'Pharmacy'
}
};
const { auth } = useAuth();
const decoded = auth?.accessToken ?
jwtDecode(auth.accessToken) : undefined;
```

```
const roles =
Object.values(decoded?.UserInfo?.roles).filter(Boolean)
|| [];
const lastElement = roles.pop();
```

```
const items = data.filter(item => item.faculty ===
lastElement)
```

```
//console.log(lastElement);
const classes = useStyles();
return (
<ThemeProvider theme={outerTheme}>
<Box
sx={{
display: "block",
// backgroundColor: "#eef0f2",
width: "100%",
height: "100%",
```

Software Testing

Unit testing

Module testing

Integration testing

System testing

Database & Acceptance