THE STATE UNIVERSITY OF ZANZIBAR

SCHOOL OF COMPUTING, COMMUNICATION AND MEDIA STUDIES

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION

TECHNOLOGY FINAL YEAR PROJECT SYSTEM DOCUMENTATION

FORMAT 2023/2024

COURSE CODE	
COURSE TITLE	FINAL YEAR PROJECT
DATE	20/08/2024
SUPERVISOR	DR. OMAR KOMBO
STUDENT NAME	ABDUL-RAHMAN SHAIBU KHEIR
YEAR	THIRD YEAR
REGISTRATION NUMBER	BCS/16/21/011/TZ

ACKNOWLEDGEMENT

Firstly, I would like to thank almighty God for empowering me and enabling his guidance to me on everything as well as giving me good health and power to accomplish my report.

I would like to express my gratitude to all my lectures at SUZA for their not only proper cooperation but also appreciation, and to many individuals who have willingly contributed to an ultimate success of my project system documentation.

I wish to express my special thanks to my beloved supervisor ***DR. OMAR KOMBO*** for his both encouragement motivation and continuous support simultaneously with his advice, through all the time. Without his support, I could not have completed this completed report.

Also, I would like to give my appreciation to THE STATE UNIVERSITY OF ZANZIBAR (SUZA) that is has been effortful in lecturing for all semesters.

ABSTRACT

This is a report concerning with **Web Development** of University Cleaners Management.

The first two weeks out of four months in my final year project system documentation format was about providing short and clear description and intention of my project.

Providing brief investigation of the system under consideration gives a clear picture of how actually the physical system is? Brief analysis or summary of the problems identified relating to the project or issues to be addressed by the project.

Explaining the scenarios of how my proposed system will be used and to articulate how this solution is better than the current practice. Describing the envisioned benefits of my proposed solution. Explaining the scope/main functions that should be covered by proposed solution Also explaining the objectives, project background and motivation.

The second two weeks out of four months in my final year project system documentation format was about methodology based on software development approach, software development life cycle model, and system architecture and software development tools.

The fifth weeks out of four months in my final year project system documentation format was about requirement analysis and modeling. Requirement Here will touch requirement determination and structuring.

The last weeks are about system design, system implementation and testing, conclusions, recommendations challenges and references.

Generally, Web Development of University Cleaners Management System will be divided into 6 chapters namely chapter one, chapter two chapter three chapter four chapter five and chapter six, however sub topics respectively.

Chapter one, introduction to University Cleaners Management System, project documentation format start and end date.

And the rest explain about methodology, requirement analysis and modeling, system design and system implementation and testing.

Table of Contents

LIST OF ABBREVIATIONS

API	APPLICATION PROGRAMMING INTERFACE
ERD	ENTITY RELATIONAL DIAGRAM
FK	FOREIGN KEY
PK	PRIMARY KEY
S/N	SERIAL NUMBERS
SLDC	SOFTWARE DEVELOPMENT LIFE CYCLE MODEL
SRS	SOFTWARE REQUIREMENT SPECIFICATION
SUZA	THE STATE UNIVERSITY OF ZANZIBAR.
UCMS	UNIVERSITY CLEANERS MANAGEMENT SYSTEM

CHAPTER 1

INTRODUCTION ABOUT THE PROJECT

DESCRIPTION OF THE PROJECT

The University Cleaners Management System (UCMS) is designed to streamline and manage cleaning services provided to various client organizations. This system facilitates the management of cleaning companies, their staff, client organizations, client sites, tools, and related complaints and attachments. It ensures efficient handling of staff complaints, tracking of tools and equipment, and effective communication between cleaning companies and client organizations.

Problem Statement

Managing cleaning services involves coordinating multiple entities such as cleaning companies, staff, client organizations, and their sites. Current manual or inefficient digital systems often lead to issues such as:

- Inefficient tracking of cleaning tasks and staff assignments.
- Poor management and resolution of staff complaints.
- Difficulty in monitoring and maintaining tools and equipment.
- Challenges in maintaining and accessing records and attachments related to complaints.

Problems, Solution, and Scope

Problems:

- **Inefficient Management:** Managing multiple entities like cleaning companies, staff, client organizations, and tools can be cumbersome without a centralized system.

- **Complaint Handling:** Staff complaints and their resolution processes are often poorly managed, leading to unresolved issues and dissatisfaction.
- **Tool Tracking:** Difficulty in tracking tools and equipment across different client sites.
- **Record Management:** Challenges in managing and accessing attachments related to complaints.

Solution: The UCMS addresses these problems by providing a comprehensive digital platform with the following features:

- **Centralized Management:** A centralized system to manage cleaning companies, staff, client organizations, and client sites.
- **Complaint Management:** An integrated system for submitting, tracking, and resolving staff complaints.
- **Tool Tracking:** Efficient tracking of tools and equipment assigned to different client sites.
- **Attachment Management:** A structured way to handle and access attachments related to complaints.

Scope: The scope of the UCMS project encompasses the planning, design, development, and implementation of a comprehensive system to manage cleaning company operations, staff, client organizations, sites, tools, and related complaints and attachments. The project aims to provide a centralized platform to streamline and enhance the management of these entities through an integrated database system.

Key Objectives

- **Comprehensive Management System:** Develop a robust system to manage cleaning companies, staff, client organizations, sites, tools, complaints, and attachments.
- **Role-Based Access Control:** Implement role-based access control to ensure data security and integrity, granting appropriate permissions based on user roles.
- **Task Management:** Incorporate task management features to assign tasks to cleaning staff, track progress, and ensure efficient task completion.
- **Operational Efficiency:** Enhance overall efficiency, transparency, and accountability in university cleaning operations, contributing to a cleaner and safer environment for all stakeholders.

Objectives

- **Streamline Operations:** Create an efficient and user-friendly system for managing cleaning services and related entities.
- **Enhance Complaint Handling:** Provide a robust mechanism for managing and resolving staff complaints.
- **Improve Tracking:** Enable effective tracking of tools and equipment across various client sites.
- **Centralize Record Management:** Centralize the management of records and attachments, making them easily accessible and manageable.

Project Background and Motivation

The significance of this project lies in bridging the gap between existing manual systems and the potential for a technology-driven approach to manage university cleaners. The motivation stems from observed deficiencies in previous attempts. By addressing the shortcomings of manual processes, this project aims to enhance operational efficiency and elevate the quality of service delivery. The overarching goal is to create a transformative solution that not only addresses existing gaps in cleaners management but also sets an instance for efficient support staff management within university settings. The identified knowledge gaps and previous studies serve as the foundation for this project's approach to effective cleaner management in academic institutions.

Managing university cleaners has traditionally relied on manual processes for crucial aspects such as registration, work area allocation, tool monitoring, and complaint management. Previous attempts have shown limited integration of technology, often resulting in inefficient management systems and inadequate resolution of challenges in accurately managing cleaner information and tasks within university settings.

Technical Feasibility

The technical feasibility of the UCMS project assesses the practicality of the proposed system based on current technology and resources. The system will utilize modern web technologies such as React for the frontend and Spring Boot for the backend, ensuring scalability, maintainability, and performance. Integration with existing university IT infrastructure will be evaluated to ensure compatibility and minimal disruption. Additionally, the project will leverage robust security protocols to protect sensitive data, ensuring compliance with industry standards.

Operational Feasibility

Operational feasibility examines the ability of the university and associated cleaning companies to effectively implement and utilize the UCMS. The system will streamline operations, enhance communication, and improve efficiency in managing cleaning tasks, staff, and complaints. Training programs will be conducted to ensure that all users, including administrators, staffs and cleaning staff, can effectively use the system. User feedback will be incorporated into the development process to ensure the system meets the operational needs of all stakeholders.

Legal Feasibility

Legal feasibility assesses the compliance of the UCMS project with relevant laws and regulations. The system will adhere to data protection and privacy laws to ensure the secure handling of personal and sensitive information. Contracts and agreements with cleaning companies and client organizations will be reviewed to ensure they align with the implementation of the new system. Any legal constraints or requirements will be identified and addressed to mitigate potential risks.

Economic Feasibility

Economic feasibility evaluates the cost-effectiveness and financial viability of the UCMS project. A detailed cost-benefit analysis will be conducted, considering the initial development costs, ongoing maintenance, and potential savings from increased efficiency and reduced manual workload. The system is expected to provide significant and improved operational efficiency, better resource management, and enhanced accountability.

CHAPTER 2

METHODOLOGY

Software Development Approach

In this project, agile development approach has been adopted as the model of software development.

At its core, Agile modeling emphasizes transparency, iteration, and empowerment. ***Transparency*** ensures that everyone involved in the project has a clear understanding of the goals, progress, and challenges. ***Iteration*** means that development occurs in small, manageable increments, allowing for rapid feedback and adaptation. Empowerment encourages teams to take ownership of their work, fostering creativity and innovation.

One of the key **advantages** of the Agile approach is its preference for face-to-face communication enabling better cooperation, welcoming changes, integrating them smoothly into ongoing development cycles and facilitating responsiveness.

However, the Agile model does have its **challenges**. For instance, in case of some software deliverable, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.

Software development life cycle model (SDLC)

SDLC: Is the process of developing, modifying, maintaining, and replacing a software system. The following are the steps for software development life cycle:

- Planning: In the planning phase, the UCMS project manager and business analyst collected requirements from stakeholders regarding how the UCMS will work and what features it requires
- Designing: The Software Requirements Specification (SRS) documents were sent to the UCMS software development team at this stage. Then, the development team carefully reviewed the document to understand client requirements.
- Development: At this stage developers involved on creating APIs required and adding various functionalities to UCMS.

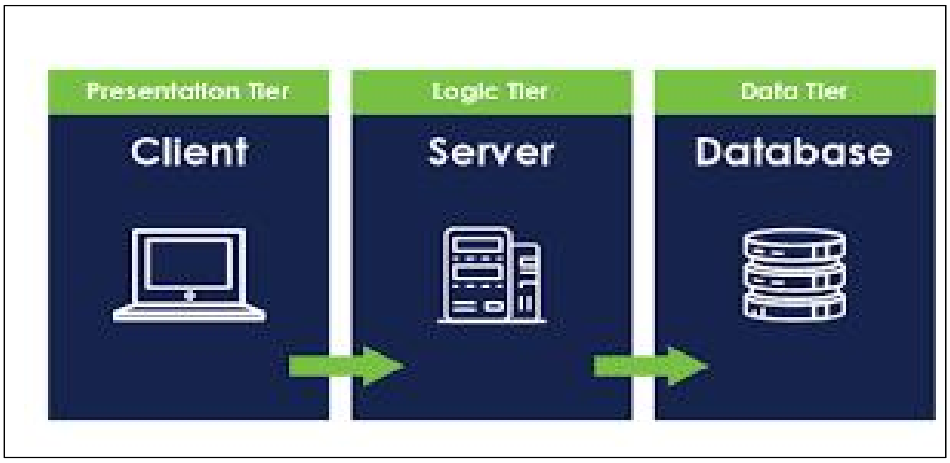
- Testing: In this step, the tester performed system testing to ensure that the UCMS is working correctly without some bugs.
- Deployment: Lastly, the developers deployed the codes and offer end-users the final version of the UCMS.

System Architecture

A system Architecture is the conceptual model that defines the structure, behavior and more views of the system. An architecture description is a formal description and representation of a system, organized in a way that support reasoning about the structures and behaviors of the system.

The system architecture of the project is **Three-Tier Architecture**

- **Presentation Tier**: A React frontend that interacts with the user.
- **Business Logic Tier**: Spring Boot backend that handles the application logic and communicates with the database.
- **Data Tier**: MySQl database that stores the application's data.



Software development tools

Software development tool: is a computer program that software developers use to create, debug, maintain, or otherwise support otherwise support other programs and applications. The following are the programming tools will used in the system:

- Programming tools
  - Front-end: JavaScript programming language.
  - Back-end: Java programming language.
- List of tools
  - Front-end: React library.
  - Back-end: Spring boot framework.
  - Database: MySQL Driver.
- IDEs
  - Front-end: Vs-code.
  - Back-end: IntelliJ IDEA Community.
- Version Control: Git for version control and collaboration

CHAPTER 3

REQUIREMENTS ANALYSIS AND MODELING

Requirements analysis is the process of determining user expectations for a new or modified product.

Requirements modeling are the process used in software development projects where requirements and solutions constantly evolve through collaborative efforts and teamwork.

Requirement determination

Involves studying the existing system and gathering details to find out what are the requirements, how it works and where improvements should be made.

Information gathering techniques

Are repeated processes that are the used to create and organize data across different kinds of sources.

During information gathering , I used the following techniques:

- Interviewing: Stakeholders, participants and experts are interviewed to identify risks.

Functional requirements

REQ ID	**REQUIREMENTS DESCRIPTION**
REQ 0001	The system shall allow staffs to register themselves
REQ 0002	The system shall allow staffs to submit their complains to company staffs's client organisation
REQ 0003	The system shall allow staffs to attach their complains to company staffs's client organisation
REQ 004	The system shall allow administrator to allocate company staffs to their client organization
REQ 005	The system shall allow administrator to allocate company staffs to the client sites from their client organization
REQ 006	The system shall allow administrator to manage tools in a client sites
REQ 007	The system shall allow staff to update their personal information
REQ 008	The system shall allow administrators to view and generate reports on

	staff complaints
REQ 009	The system shall allow staff to upload and manage attachments related to their complaints.
REQ 010	The system shall allow administrators to manage the list of client organizations and their details.
REQ 011	The system shall allow administrators to manage the list of client sites and their details.

Non – functional requirements

This defines system attributes such as security, reliability, performance, maintainability, scalability and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

REQ ID	**REQUIREMENTS DESCRIPTION**
REQ 0001	Scalability: The system must be scalable enough to support 100000 visits at the same time while maintaining optimal performance.
REQ 0002	Portability: A program running on windows must be able to run on UNIX without any change in its behavior and performance.
REQ 0003	Reliability: The system must perform without failure in 95 percent of use cases during a month.
REQ 0004	Availability: The web interface must be available 99.98 percent of the time every month during a day.
REQ 0005	Usability: The error rate of users submits their cases evidence must not exceed 10 percent.
REQ 0006	Security: The password must be encrypted.

Requirement Structuring

It is a hierarchical, usually tree-shaped description of all project requirements which must be present in the end product in order to deliver the expected business value.

Process Modelling

It is a framework used by cognitive psychologists to explain and describe the processes of the human brain

OBJECT ORIENTED APPROACH

Use-case diagram to illustrate how current system is implemented.

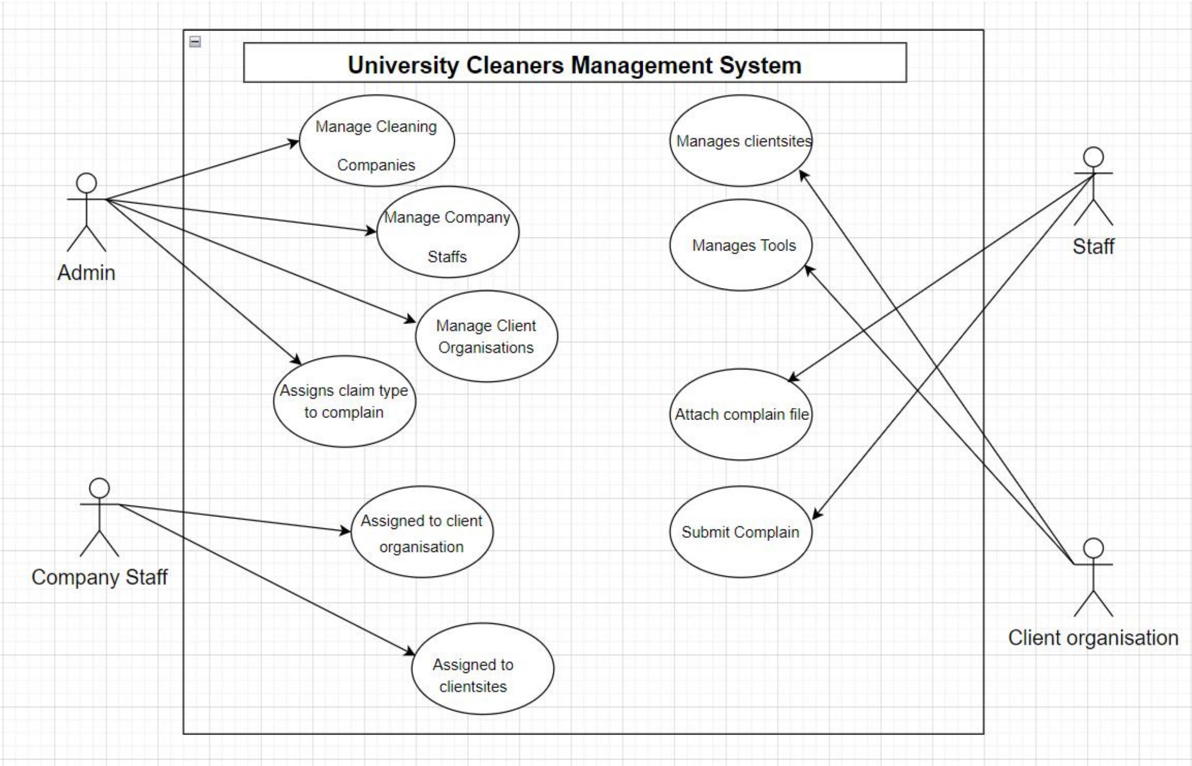
A *usecase diagram* is a graphical depiction of a user's possible interactions with a system.

Use-case model (use-case diagram and use-case description) to illustrate how proposed system is implemented

The Use-case model is defined as a model which is used to show how users interact with the system in order to solve a problem.

The use-case model defines the user's objective, the interactions between the system and the user, and the system's behavior required to meet these objectives.

Use-case diagram



Use-Case Description

S/N	ACTOR	DESCRIPTIONS
1	Staff	This is a staff from the client site who can either submit or attach complain
2	Client organisation	This is an organisation from a cleaning company that is responsible for managing both tools and client sites

3	Company staff	This is a company staff from cleaning company who is assigned to client organisation as well as client sites
4	Admin	Is the one who is responsible for managing cleaning company,company staffs, client organisations and Assigning complain type

Data Modelling

Data Modelling is the process of creating a data model for an information system by applying certain formal techniques.

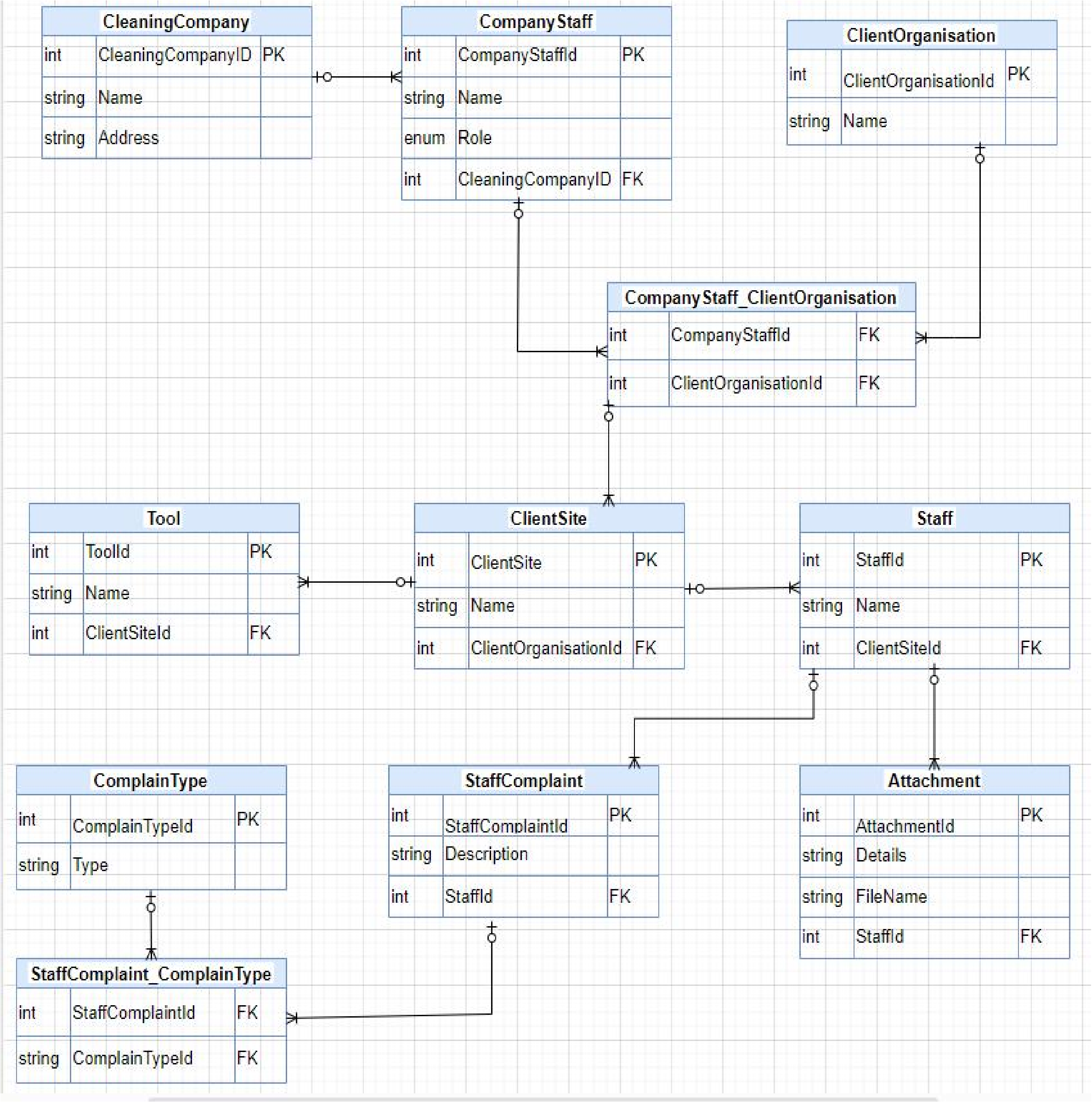
OBJECT ORIENTED APPROACH

Class diagram

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations and the relationships among the objects.

Entity relationship diagram

Is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology system.



CleaningCompany		
int	CleaningCompanyID	PK
string	Name	
string	Address	

Company Staff		
int	CompanyStaffId	PK
string	Name	
enum	Role	
int	CleaningCompanyID	FK

ClientOrganisation		
int	ClientOrganisationId	PK
string	Name	

Company Staff_ClientOrganisation		
int	CompanyStaffId	FK
int	ClientOrganisationId	FK

Tool		
int	ToolId	PK
string	Name	
int	ClientSiteId	FK

ClientSite		
int	ClientSite	PK
string	Name	
int	ClientOrganisationId	FK

Staff		
int	StaffId	PK
string	Name	
int	ClientSiteId	FK

ComplainType		
int	ComplainTypeId	PK
string	Type	

StaffComplaint		
int	StaffComplaintId	PK
string	Description	
int	StaffId	FK

Attachment		
int	AttachmentId	PK
string	Details	
string	FileName	
int	StaffId	FK

StaffComplaint_ComplainType		
int	StaffComplaintId	FK
string	ComplainTypeId	FK

CHAPTER 4

System Design

Architectural design: a description of the architecture of the program

Is the process of defining a collection of hardware and software components and their interface to establish the framework for the development of a computer system. This helps stakeholders to understand and analyze how the system will achieve essential qualities such as modifiability, availability and security.

Database Design

Is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model.

Relational Model

Is an approach to managing data using a structure and language consistent with firstorder predicate logic. In this model, the data is organized into a collection of twodimensional inter-related tables, also known as relations. Each relation is collection of columns and rows, where the column represents the attribute of an entity and the rows represent the records. This data model provides easy data sorting and data sorting and data access.

Data Description

Table Name: The name for the Table.

Column Name: The predefined name for the column.

Description: A detailed description of the contents and purpose for the column

Data Type & Size: The predefined characteristics for the column.

Table Name: Cleaning company

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Company name	VARCHAR

Table Name: Company staff

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Staff name	VARCHAR
3	role	Staff role	VARCHAR
4	cleaning_company_id	Foreign key to cleaning_company	BIGINT

Table Name: Client organisation

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Organisation name	VARCHAR

Table Name: Client site

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Site name	VARCHAR
3	client_organisation_id	Foreign key to client_organisation	BIGINT

Table Name: Tool

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Tool name	VARCHAR
3	client_site_id	Foreign Key to oclient_site	BIGINT

Table Name: Staff

S/N	COLUMN NAME	DESCRIPTION	DATA TYPE
1	id	Primary key	BIGINT
2	name	Tool name	VARCHAR

3	client_site_id	Foreign Key to oclient_site	BIGINT

Table Name: Staff complain

S/N	**COLUMN NAME**	**DESCRIPTION**	**DATA TYPE**
1	id	Primary key	BIGINT
2	description	Complain description	VARCHAR
3	staff_id	Foreign Key to staff	BIGINT

Table Name: Attachment

S/N	**COLUMN NAME**	**DESCRIPTION**	**DATA TYPE**
1	id	Primary key	BIGINT
2	name	Tool name	VARCHAR
3	staff_id	Foreign Key to staff	BIGINT

Table Name: Complain type

S/N	**COLUMN NAME**	**DESCRIPTION**	**DATA TYPE**
1	id	Primary key	BIGINT
2	type	Complain type	VARCHAR

Table Name: Company-staff_Client-organisation

S/N	**COLUMN NAME**	**DESCRIPTION**	**DATA TYPE**
1	Company_staff_id	Foreign key to company_staff	BIGINT
2	Client_organisation_id	Foreign Key to client_organisation	BIGINT

Table Name: Staff-complain_Complain-type

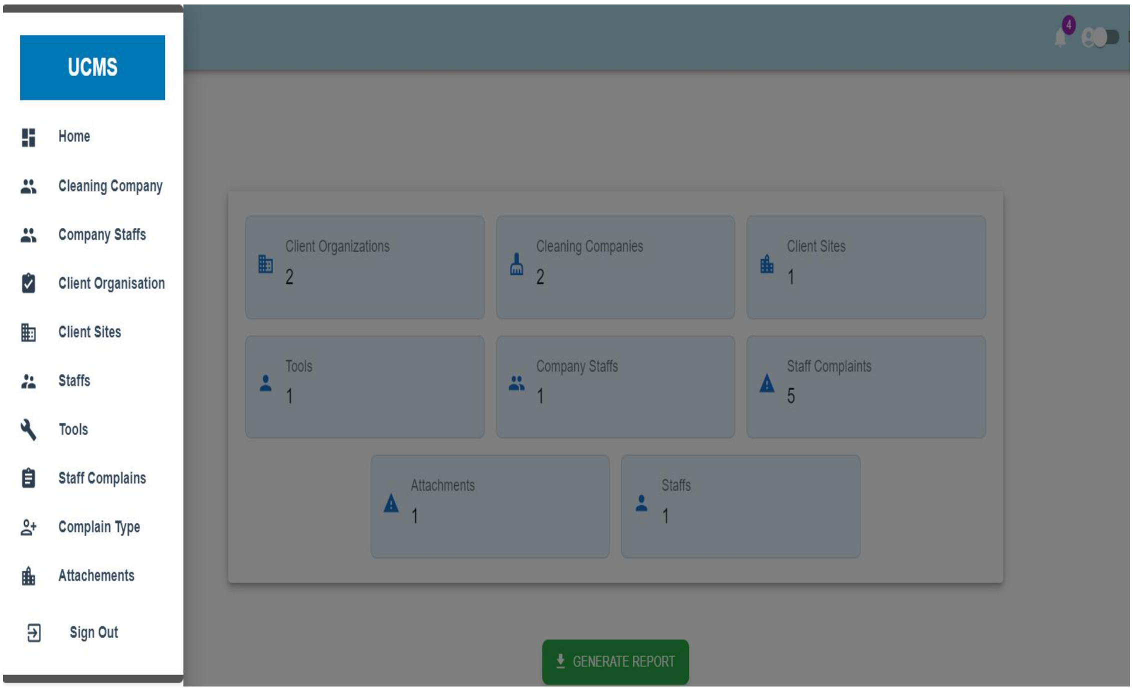
S/N	**COLUMN NAME**	**DESCRIPTION**	**DATA TYPE**
1	Staff_complain_id	Foreign key to staff_complain	BIGINT
2	Client_organisation_id	Foreign key to complain_type	BIGINT

User Interface Design

Is the design of user interfaces for machines and software such as computers, mobile devices and other electronic devices with the focus on maximizing usability and the user experience.

Interface design sample

Here focuses on expecting what users might need to do and ensuring that the interface has elements that are easy to access, understand and use to facilitate those actions.



Client Organizations 2

Cleaning Companies 2

Client Sites 1

Tools 1

Company Staffs 1

Staff Complaints 5

Attachments 1

Staffs 1

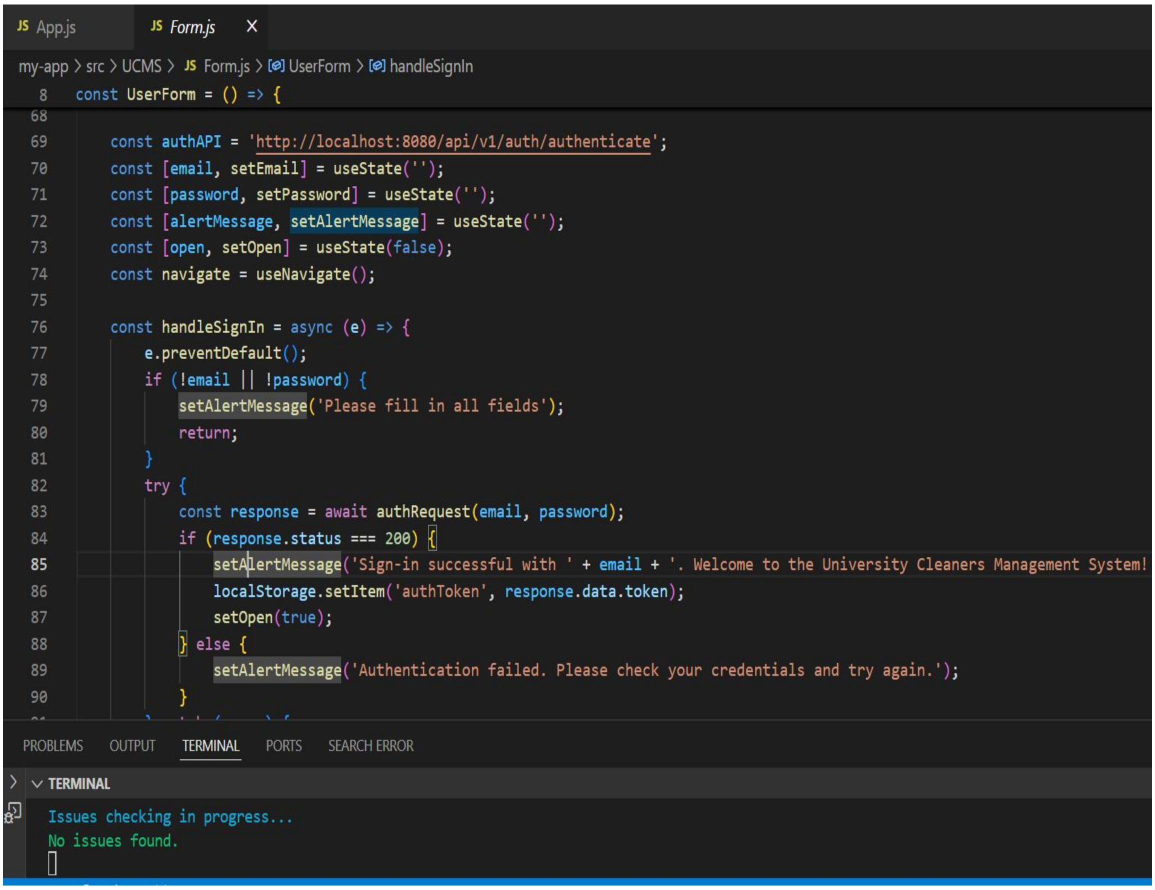
CHAPTER 5

System implementation and testing

Technologies

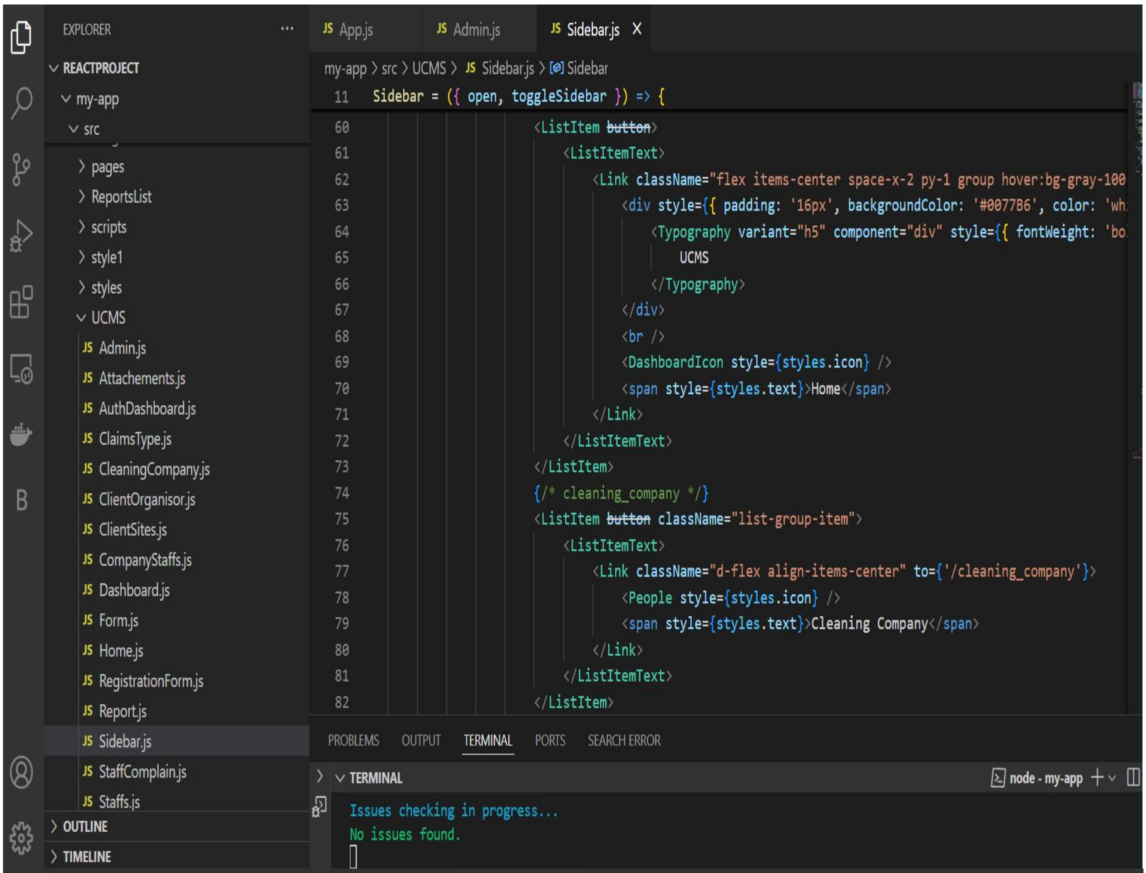
A general term covering the development methods, programming languages and tools to support them that may be used in the development of software. The technologies I will use in my system will into two categories, frontend and backend. For frontend I will use **React library** with JavaScript and **Spring Boot** for backend

REACT



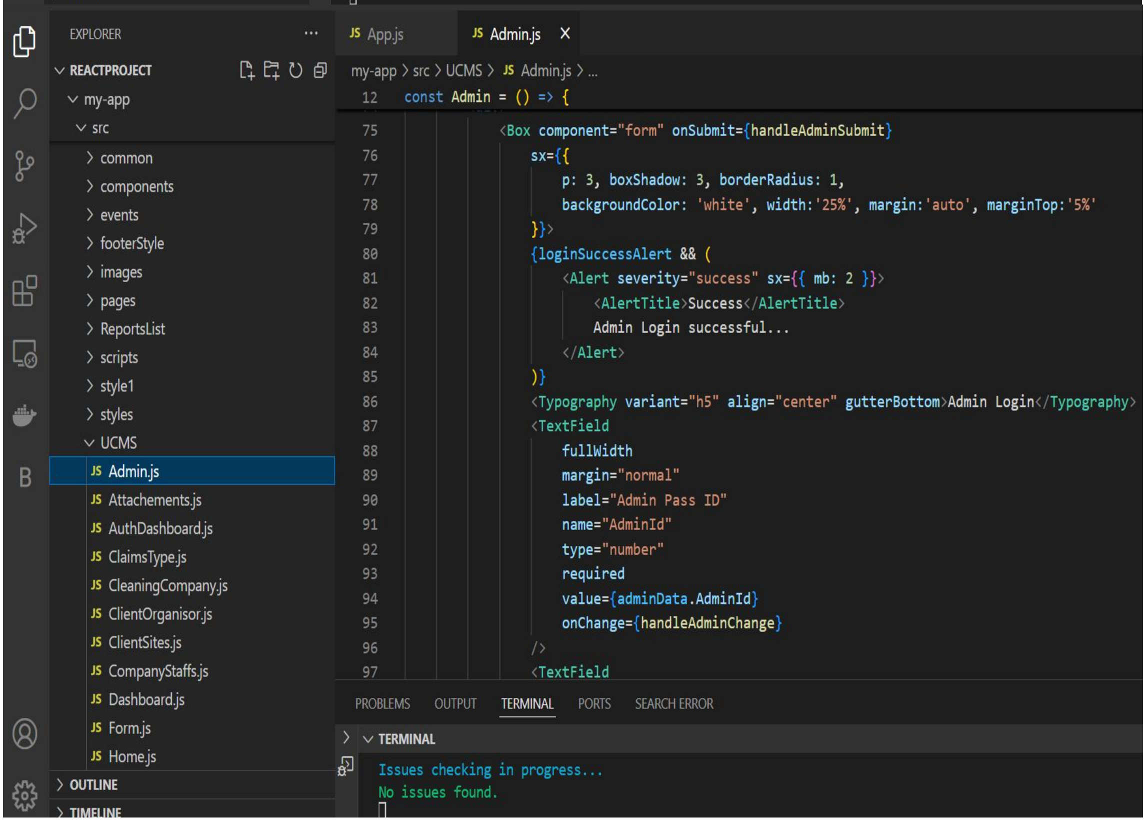
```
const UserForm = () => {

    const authAPI = 'http://localhost:8080/api/v1/auth/authenticate';
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [alertMessage, setAlertMessage] = useState('');
    const [open, setOpen] = useState(false);
    const navigate = useNavigate();

    const handleSignIn = async (e) => {
        e.preventDefault();
        if (!email || !password) {
            setAlertMessage('Please fill in all fields');
            return;
        }
        try {
            const response = await authRequest(email, password);
            if (response.status === 200) {
                setAlertMessage('Sign-in successful with ' + email + '. Welcome to the University Cleaners Management System!
                localStorage.setItem('authToken', response.data.token);
                setOpen(true);
            } else {
                setAlertMessage('Authentication failed. Please check your credentials and try again.');
            }
```

```
Issues checking in progress...
No issues found.
```



```
Sidebar = ({ open, toggleSidebar }) => {
            <ListItem ~~button~~>
                <ListItemText>
                    <Link className="flex items-center space-x-2 py-1 group hover:bg-gray-100
                        <div style={{ padding: '16px', backgroundColor: '#0077B6', color: 'wh
                            <Typography variant="h5" component="div" style={{ fontWeight: 'bo
                                UCMS
                            </Typography>
                        </div>
                        <br />
                        <DashboardIcon style={styles.icon} />
                        <span style={styles.text}>Home</span>
                    </Link>
                </ListItemText>
            </ListItem>
            {/* cleaning_company */}
            <ListItem ~~button~~ className="list-group-item">
                <ListItemText>
                    <Link className="d-flex align-items-center" to={'/cleaning_company'}>
                        <People style={styles.icon} />
                        <span style={styles.text}>Cleaning Company</span>
                    </Link>
                </ListItemText>
            </ListItem>
```

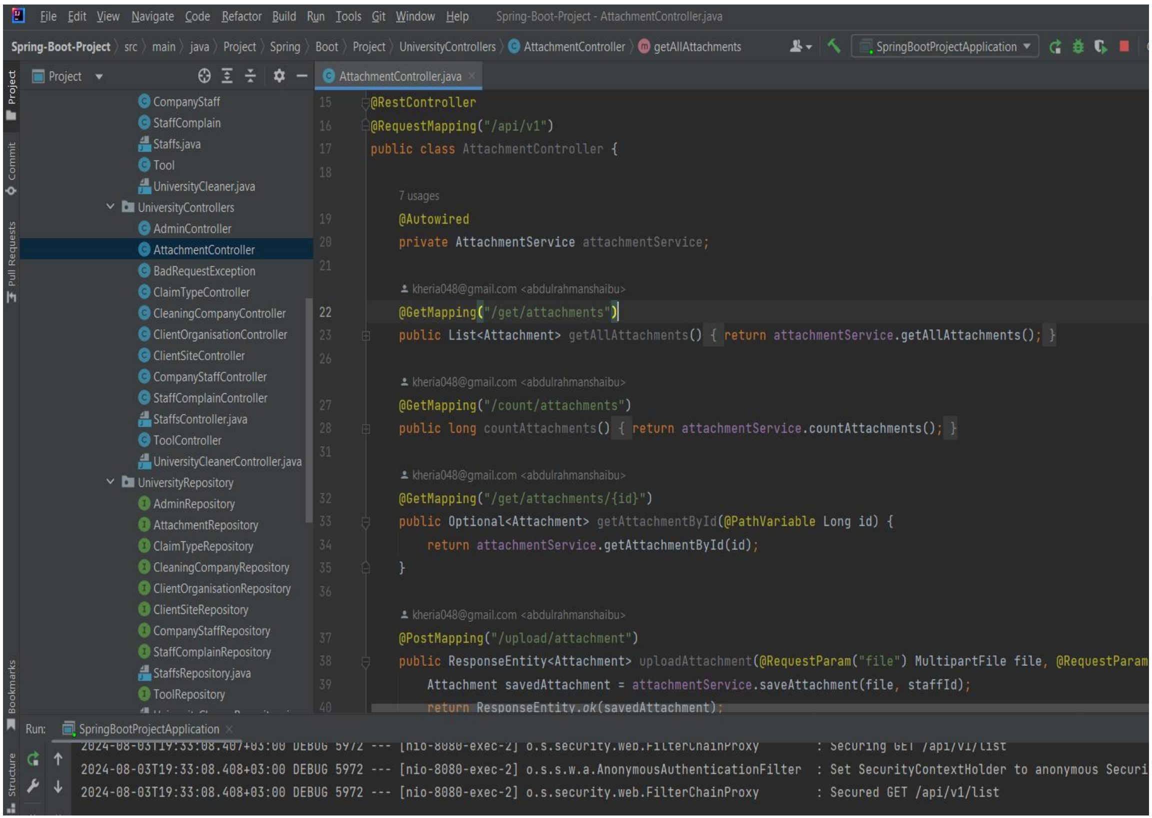
```
Issues checking in progress...
No issues found.
```



```
const Admin = () => {
            <Box component="form" onSubmit={handleAdminSubmit}
                sx={{
                    p: 3, boxShadow: 3, borderRadius: 1,
                    backgroundColor: 'white', width:'25%', margin:'auto', marginTop:'5%'
                }}>
                {loginSuccessAlert && (
                    <Alert severity="success" sx={{ mb: 2 }}>
                        <AlertTitle>Success</AlertTitle>
                        Admin Login successful...
                    </Alert>
                )}
                <Typography variant="h5" align="center" gutterBottom>Admin Login</Typography>
                <TextField
                    fullWidth
                    margin="normal"
                    label="Admin Pass ID"
                    name="AdminId"
                    type="number"
                    required
                    value={adminData.AdminId}
                    onChange={handleAdminChange}
                />
                <TextField
```

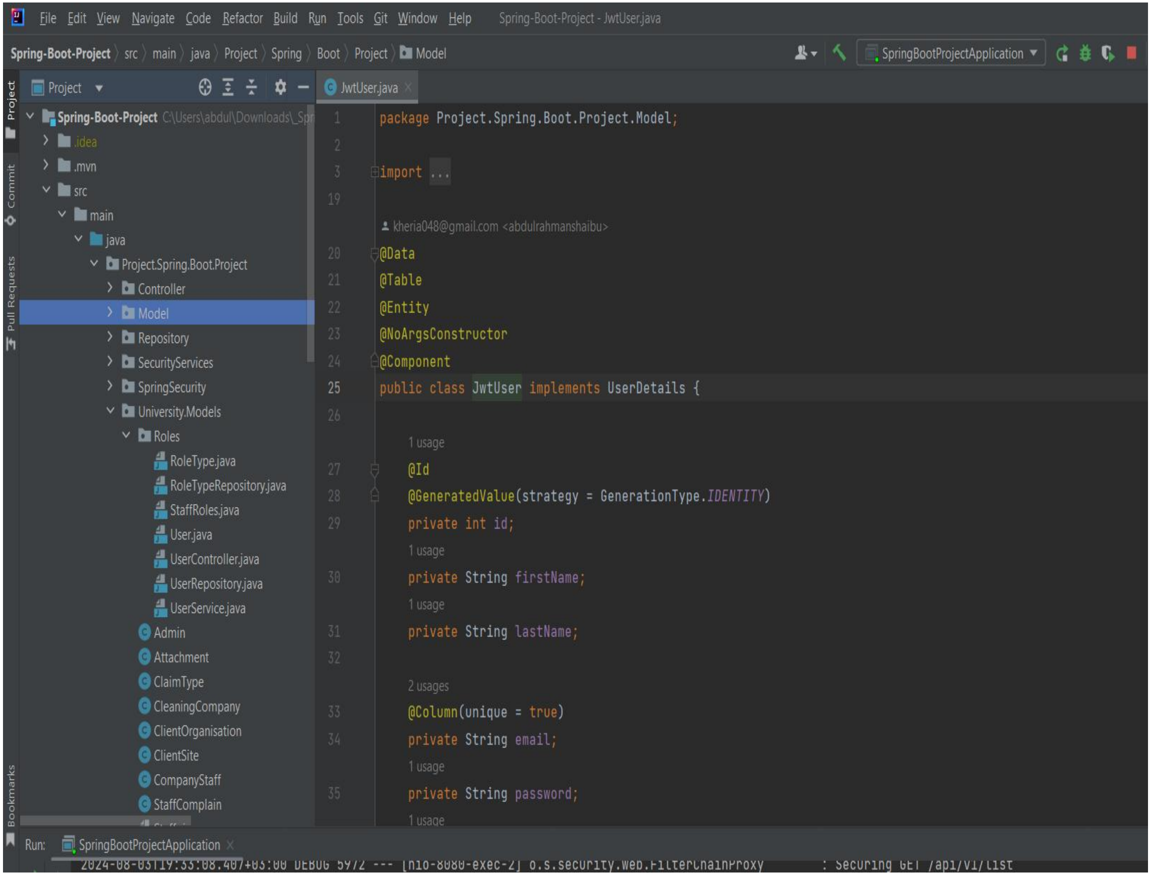
```
Issues checking in progress...
No issues found.
```

SPRINGBOOT



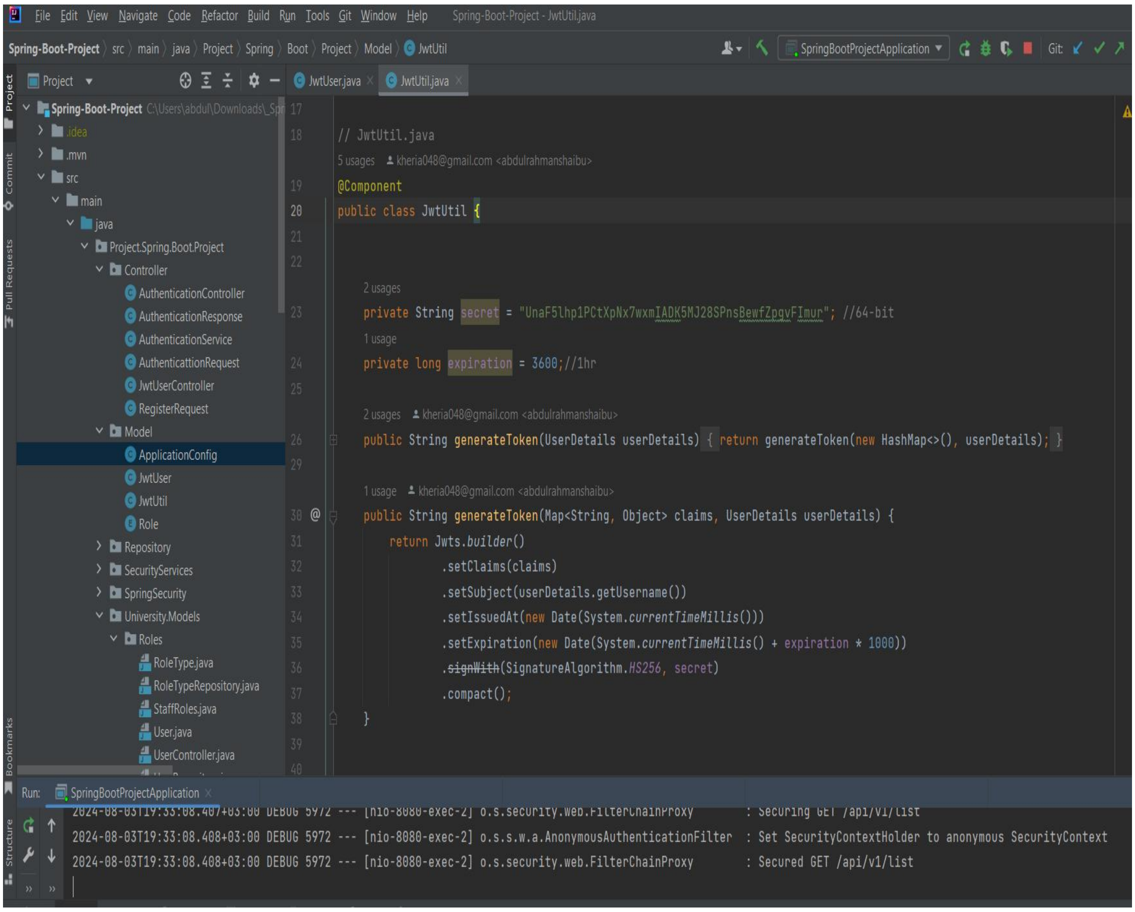
```java
@RestController
@RequestMapping("/api/v1")
public class AttachmentController {

    7 usages
    @Autowired
    private AttachmentService attachmentService;

    kheria048@gmail.com <abdulrahmanshaibu>
    @GetMapping("/get/attachments")
    public List<Attachment> getAllAttachments() { return attachmentService.getAllAttachments(); }

    kheria048@gmail.com <abdulrahmanshaibu>
    @GetMapping("/count/attachments")
    public long countAttachments() { return attachmentService.countAttachments(); }

    kheria048@gmail.com <abdulrahmanshaibu>
    @GetMapping("/get/attachments/{id}")
    public Optional<Attachment> getAttachmentById(@PathVariable Long id) {
        return attachmentService.getAttachmentById(id);
    }

    kheria048@gmail.com <abdulrahmanshaibu>
    @PostMapping("/upload/attachment")
    public ResponseEntity<Attachment> uploadAttachment(@RequestParam("file") MultipartFile file, @RequestParam
        Attachment savedAttachment = attachmentService.saveAttachment(file, staffId);
        return ResponseEntity.ok(savedAttachment);
```

```
2024-08-03T19:33:08.407+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy        : Securing GET /api/v1/list
2024-08-03T19:33:08.408+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous Securi
2024-08-03T19:33:08.408+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy        : Secured GET /api/v1/list
```



```java
package Project.Spring.Boot.Project.Model;

import ...

kheria048@gmail.com <abdulrahmanshaibu>
@Data
@Table
@Entity
@NoArgsConstructor
@Component
public class JwtUser implements UserDetails {

    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    1 usage
    private String firstName;
    1 usage
    private String lastName;

    2 usages
    @Column(unique = true)
    private String email;
    1 usage
    private String password;
    1 usage
```

```
2024-08-03T19:33:08.407+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy        : Securing GET /api/v1/list
```

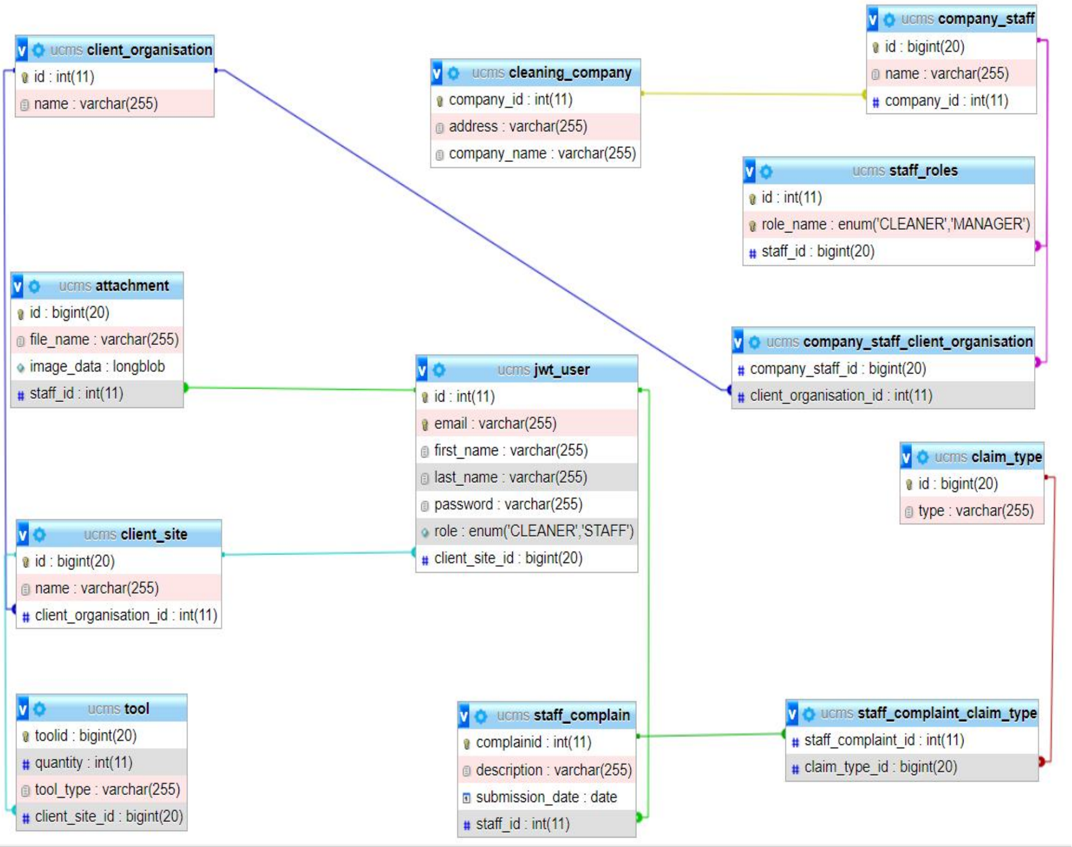


```
// JwtUtil.java
5 usages  kheria048@gmail.com <abdulrahmanshaibu>
@Component
public class JwtUtil {


    2 usages
    private String secret = "UnaF5lhp1PCtXpNx7wxmIADK5MJ28SPnsBewfZpqvFImur"; //64-bit
    1 usage
    private long expiration = 3600;//1hr

    2 usages  kheria048@gmail.com <abdulrahmanshaibu>
    public String generateToken(UserDetails userDetails) { return generateToken(new HashMap<>(), userDetails); }

    1 usage  kheria048@gmail.com <abdulrahmanshaibu>
    public String generateToken(Map<String, Object> claims, UserDetails userDetails) {
        return Jwts.builder()
                .setClaims(claims)
                .setSubject(userDetails.getUsername())
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + expiration * 1000))
                .signWith(SignatureAlgorithm.HS256, secret)
                .compact();
    }
```

```
2024-08-03T19:33:08.407+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy        : Securing GET /api/v1/list
2024-08-03T19:33:08.408+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous SecurityContext
2024-08-03T19:33:08.408+03:00 DEBUG 5972 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy        : Secured GET /api/v1/list
```

Database implementation

A database implementation is the process of installing database software, performing, configuration and customization, running and testing the database and then integrating it with applications. Finally, the implementation process involves training the users.



Data Dictionary

This is the collection of names, definitions and attributes about data elements that are being used or captured in a database, information system, or a part of a research project.

Internal Schema of database (database schema)

Database schema: Is the structure of a database described in a formal language supported by the database management system. The term schema refers to the organization of data as a blueprint of how the database is constructed.
The internal schema is also known as physical schema. It uses the physical data model. It is used to define that how the data will be stored in a block.

cleaning_company (id(pk), name)
company_staff (id(pk), name, role, cleaning_company_id(fk))
client_organisation (id(pk), name)
client_site (id(pk), name, client_organisation_id(fk))
tool (id(pk), name, client_site_id(fk))
staff (id(pk), name, client_site_id(fk))
staff_complain (id(pk), description, staff_id(fk))
attachment (id(pk), file_name, staff_id(fk))
complain_type (id(pk), type)
company_staff_client_organisation(company_staff_id(fk),client_organisation_idk))
staff_complaint_complain_type (staff_complaint_id(fk), complain_type_id(fk)

TESTING

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. Takes as input, all of the integrated components that have passed integration testing.

- Yes, each functionality of the system produce an expected output because, the system contains all the functional specifications which form a basis for the entire test to be conducted. And contains the information about how the system will be perceived from a business process perspective.
- Yes, the system responds with invalid data entry, when you enter an invalid value or do not enter a value for a required field.
- Yes, the system responds with incorrect login credentials and this refers to your login details; email address and password when one among of these or both has entered incorrect into login form. Also may happen when account is locked out.

- Yes, my system restricts with user to perform non-privileged functionalities, non-privileged users can only modify the report form while super-user can change password, login to the system to view all reported cases and to work with. All are done by using enhanced security in order to prevent users from choosing their own passwords and get access as super-user.
- Yes, I have implemented the system using the proposed techniques and development approaches which is agile development approach. Where, agile modelling is a methodology for modeling and documenting software systems based on best practices.

REPORTING TESTING

User Interfaces

Is an interface where a user interacts with the website they’re using. It provides a pictorial representation of most important user interfaces of the main functionalities of the system.

Strength and Limitation of the system

What is covered from requirements

In this project, most of the requirement as mentioned above include functional requirements and non-functional requirements have been covered.

What is not covered

I have not covered the part of providing feedback for the complains that have been submitted by the staff in a client site

CHAPTER 6

Conclusion, Recommendations, Challenges, and References

Conclusion

The University Cleaners Management System (UCMS) facilitates seamless integration of cleaning services by fostering relationships among cleaning companies, client organizations, and client sites. Additionally, it enhances the management and resolution of complaints from staff regarding the cleaning services provided by the organization. The UCMS ensures efficient communication and coordination, thereby improving the overall quality of cleaning services and satisfaction levels among stakeholders.

Recommendations

It is recommended that client organizations prioritize the resolution of complaints submitted by staff at client sites. This proactive approach will significantly enhance the quality of cleaning services and adoptive positive work environment. Addressing staff concerns promptly and effectively is essential for maintaining high standards of cleanliness and operational efficiency.

Challenges

Challenges are an inevitable part of any project. Throughout the development of the UCMS, several challenges were encountered:

- **Integration Complexity**: Coordinating backend and frontend systems required careful planning to ensure seamless communication between all components.
- **Data Management**: Managing large volumes of data on cleaning schedules, staff, and complaints demanded strict accuracy and consistency.
- **Customization and Scalability**: Creating a flexible and scalable system to meet diverse client needs involved balancing customization with functionality.
- **Security and Privacy**: Ensuring the protection of sensitive data required implementing strong security measures to prevent breaches and unauthorized access.
- **User Adoption**: Transitioning users to the new system required thorough training and support to ease the change from existing processes.

References