# 

|  |  |
| --- | --- |
| NAMES | STUDENT NUMBERS |
| REEKELELITSOE TS’OAELI | 901017513 |
| REFILOE MATS’OELE | 901017318 |
| MOTLATSI TS’OENE | 901017345 |
| MPINE MOETI | 901017527 |
| MASOABI FASO | 901017170 |
| LIMPHO MOROPANE | 901017364 |

# CLASS: BSCITY2S2

# MODULE: DATABASE SYSTEMS

# TITLE: GROUP ASSIGNMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **STUDENT** | **PRIMARY ROLE** | **KEY TASKS** | **REPORT SECTIONS** |
| Reekelelitsoe Ts’oaeli | Project Manager/Features Coordinator | Manage timeline, coordinate features, ensure integration, oversee user acceptance testing, lead stakeholder communication and presentation, create feature documentation and user guide | System Overview, Key Features, Benefits and Lessons Learned (UAT insights, user guide), Conclusion (presentation feedback) |
| Refiloe Mats’oele | Frontend Development Lead | Build React.js interface, integrate APIs | FRONTEND: React.js |
| Motlatsi Ts’oene | Security and Testing Lead | Implement security (JWT, RBAC), test system | Security Measures |
| Mpine Moeti | Report Writing Lead | Write abstract, introduction, conclusion, compile report | Abstract, Introduction, Body, Conclusion, References |
| Masoabi Faso | Backend Development Lead | Develop Node.js/Express.js APIs, backend logic | BACKEND (Database: MongoDB) |
| Limpho Moropane | Database Design Lead | Design MongoDB collections, optimize queries | BACKEND (Database: MongoDB) |

CONTENTS

[Heading 5](#_Toc197442635)

[Abstract 5](#_Toc197442636)

[KEYWORDS 5](#_Toc197442638)

[1. Introduction 5](#_Toc197442639)

[2. Body 5](#_Toc197442640)

[2.1 System Overview 5](#_Toc197442641)

[2.2 Key Features 6](#_Toc197442642)

[2.3 FRONTEND: React.js 6](#_Toc197442643)

[2.4 BACKEND (Database: MongoDB) 7](#_Toc197442644)

[2.5 Security Measures 7](#_Toc197442645)

[2.6 Benefits and Lessons Learned 8](#_Toc197442646)

[3. Conclusion 8](#_Toc197442647)

[REFERENCES 9](#_Toc197442648)

# 

# DESIGN AND DEVELOPMENT OF A CREDIT BUREAU MANAGEMENT SYSTEM USING MONGODB AND REACT.JS

# Abstract

### Lenders today need accurate credit information to make good decisions. Our project is about building a Credit Bureau Management System (CBMS) that stores and manages credit records. We used MongoDB for the database and React.js for the user interface. The system helps lenders check people’s creditworthiness by tracking their loans and payments. Our system can handle a lot of data and is easy to use. In this report, we explain the design, technologies, and how the system works. We also show how it can help reduce loan defaults and improve lending practices.

KEYWORDS***:*** *Credit Bureau, MongoDB, React.js, Consumer Credit, Creditworthiness, Credit Reports, Risk Assessment, Loan Default Prevention, Full-Stack Development, Financial Systems*

# 1. Introduction

Credit bureaus collect credit data to help banks and lenders make loan decisions. They help promote safe borrowing and lending. We created a Credit Bureau Management System (CBMS) using modern web technologies for our project. We chose React.js for building a smooth, interactive frontend and MongoDB to store different types of credit records. Our system lets lenders view credit reports, check payment history, and assess risk before approving loans. This project helped us practice full-stack development, build skills in React.js and MongoDB, and understand how credit data is managed in real life. In this report, we explain our system, the challenges we met, and its value in finance.

# 2. Body

## 2.1 System Overview

Our system is built using a two-layer structure:

* **Frontend:** Built with React.js to provide an easy-to-use interface for users like lenders and credit officers.
* **Backend:** We used Node.js and Express.js to create APIs that connect the frontend to the database. **The database**
* MongoDB is our main database, which stores all the credit records in a flexible and scalable way.

This structure helped us separate the user interface from the data logic and made our system easier to manage and expand in the future.

## 2.2 Key Features

The main features we implemented include:

* **Manage Credit Records:** Users can add, update, and delete information about people’s loans, repayments, and credit scores.
* **Generate Credit Reports:** The system can create detailed reports showing a person’s borrowing history.
* **Risk Assessment:** By looking at past records, the system helps lenders identify high-risk borrowers.
* **Search and Filter:** Users can quickly search for credit profiles by name, ID number, or credit score.
* **Secure Access:** Only authorized users can log in and manage sensitive credit information.

## 2.3 FRONTEND: React.js

#### **React.js** allows us to build a dynamic, component-based interface. Some of the main pages we created are:

* **Login and Signup Pages:** To control user access.
* **Dashboard:** Shows summaries like the number of active credit records and average credit scores.
* **Credit Profile Page:** Displays detailed information about individual borrowers.

We also used React’s state management to make sure data updates in real-time without reloading pages.

## 2.4 BACKEND (Database: MongoDB)

**MongoDB** was a good fit because it can handle flexible data structures. For example, one borrower might have multiple loans, each with different payment histories. This is easy to store in MongoDB without needing complex table joins.

We created collections such as:

* **Users:** To store login information and user roles.
* **CreditRecords:** To store borrower details like loan amount, repayment history, and credit scores.

Here’s a sample credit record we stored:

json

CopyEdit

{

"borrowerName": "Jane Doe",

"loanAmount": 10000,

"repaymentHistory": [

{"date": "2025-01-01", "status": "Paid"},

{"date": "2025-02-01", "status": "Missed"}

],

"creditScore": 720

}

## 2.5 Security Measures

Since credit data is sensitive, we made sure to include some security features:

* **Login Authentication:** Using JWT tokens so only logged-in users can access the system.
* **Role-Based Access Control:** Different users have different permissions (e.g., admin vs. credit officer).
* **Encrypted Data:** We used secure connections (HTTPS) and ensured data is protected in the database.

## 2.6 Benefits and Lessons Learned

Our system offers several advantages:

* **Better Lending Decisions:** Lenders can check credit reports before approving loans.
* **Reduced Risk:** By avoiding high-risk borrowers, lenders can prevent loan defaults.
* **User-Friendly:** The system is easy to use, even for people who are not tech-savvy.

As students, we also learned valuable lessons about:

* Full-stack development using React, Node.js, and MongoDB.
* How real-world credit systems work.
* The importance of data privacy and security.

# 3. Conclusion

Through this project, our team successfully developed a working prototype of a Credit Bureau Management System using React.js and MongoDB. The system makes it easier for lenders to collect and review borrowers’ credit information, leading to smarter and safer lending decisions. We believe our project could be expanded further by adding features like automated credit scoring using machine learning or integrating with national ID systems for better verification. This project not only helped us improve our technical skills but also gave us practical insight into how financial systems operate. Moving forward, we hope to enhance the system by adding more security layers and expanding its features to handle more complex credit data.

# REFERENCES

[1] F. I. Onifade, "Credit scoring and risk assessment," \*African J. Econ. Policy\*, vol. 27, no. 2, pp. 54–70, 2020.

[10] M. Hartl, \*Learn Enough JavaScript to Be Dangerous\*. Softcover.io, 2021.

[2] T. Chen, "An overview of MongoDB for big data," \*J. Comput. Commun.\*, vol. 9, no. 1, pp. 33–39, 2021.

[3] P. K. Kannan and H. A. Li, "Digital marketing: A framework, review and research agenda," \*Int. J. Res. Marketing\*, vol. 34, no. 1, pp. 22–45, 2017.

[4] S. Das and J. Chen, "Credit risk modeling with machine learning," \*Proc. IEEE\*, vol. 108, no. 3, pp. 355–372, Mar. 2020.

[5] M. Fowler, \*NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence\*. Boston, MA, USA: Addison-Wesley, 2012.

[6] E. Evans, \*Domain-Driven Design: Tackling Complexity in the Heart of Software\*. Boston, MA, USA: Addison-Wesley, 2003.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," \*Commun. ACM\*, vol. 51, no. 1, pp. 107–113, 2008.

[8] A. P. Pavan, "Secure storage and access control in MongoDB," \*Int. J. Comput. Appl.\*, vol. 183, no. 22, pp. 1–5, 2021.

[9] D. Crockford, \*JavaScript: The Good Parts\*. Sebastopol, CA, USA: O'Reilly Media, 2008.