WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

BACHELOR THESIS

SUPERVISOR:
Asistent Dr. Florin Rosu

GRADUATE:
Serban Ples

TIMIŞOARA
2025

WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

Cloud Class

SUPERVISOR:
Asistent Dr. Florin Rosu

GRADUATE:
Serban Ples

TIMIŞOARA
2025

Abstract

Contents

Chapter 1

Introduction

1.1 Motivation

In the modern educational environment, students face a lot of challenges that can impact their ability to maximize their academic potential. With the increasing demand of coursework, assignments, and exams, it is essential for students to have access to tools that offer them an easier learning process. One of the most significant obstacles students encounter is managing study materials. It is common for students to accumulate a lot of notes, articles, and resources, but often struggle to keep them organized and accessible when needed.

In addition to document management, collaboration plays a vital role in the learning experience. Peer discussions, group study sessions, and collaborative learning are essential components of academic success. However, the logistics of coordinating these activities, especially in today's digital age, can often prove to be a lot harder than expected. Scheduling study sessions, coordinating group discussions, and sharing feedback can become disorganized without an effective system in place.

Recognizing these challenges, the development of a web application aimed at improving the student experience has become increasingly important. This application serves as a one-stop solution for students to share learning documents, create chat groups for collaboration, and schedule study sessions. By providing a platform that integrates these essential tools, students getting the support they desperately need to enhance their productivity, engage in meaningful discussions, and collaborate more efficiently. This thesis explores the creation, functionality, and potential impact of this web application on student learning, aiming to provide an innovative solution that addresses the evolving needs of today's learners.

1.2 Objectives

The primary objective of this thesis is to design and develop a web application that addresses the key challenges students face in managing learning materials, collaborating with peers, and organizing study sessions. The objectives are as follows:

1. **Document Management**: To provide a user-friendly platform where students can easily upload, share, and organize learning documents such as lecture notes, textbooks, and articles. This functionality aims to ensure that

students have quick access to the materials they need for studying and can collaborate more effectively.

2. **Collaborative Features**: To create an interactive environment for students to communicate and collaborate through chat groups. These groups will allow students to engage in discussions, share ideas, clarify doubts, and foster a sense of community and teamwork, which is vital in the learning process.

3. **Study Session Scheduling**: To incorporate a feature that enables students to schedule and manage study sessions. The scheduling tool will allow students to coordinate group study times, set reminders, and track upcoming sessions, making it easier to maintain a consistent study routine.

4. **User Experience and Accessibility**: To ensure that the application is intuitive, easy to navigate, and accessible across different devices, providing a seamless experience for students. The user interface (UI) will be designed to accommodate a wide range of users, including those with limited technical expertise.

By achieving these objectives, the web application aims to enhance the overall learning experience for students, promoting better organization, communication, and academic success.

1.3 Contextualization

In the digital age, educational systems are rapidly evolving, influenced by advancements in technology and changes in student learning behavior. Traditional study methods, such as in-person lectures and physical textbooks, have become replaced by digital tools that allow for greater flexibility and accessibility. Despite the growing presence of online resources, students still face significant challenges in organizing and accessing learning materials, collaborating with peers, and effectively managing their time.

One key issue is the fragmentation of learning resources. With a variety of platforms and tools available for storing and sharing documents, students often struggle to find a centralized place to access and manage all their study materials. Whether it's lecture notes, reading materials, or research articles, organizing these documents can be a time-consuming task, leading to inefficiency and disorganization in students' study habits.

Moreover, the collaborative aspect of learning is often hindered by communication barriers. While students rely on chat platforms and messaging apps for group discussions, the lack of integration with academic tools makes it difficult to focus on study-related tasks. Similarly, coordinating group study sessions often requires juggling multiple apps and platforms, leading to missed opportunities for productive collaboration.

The contextualization of this research emphasizes the need for a unified platform where students can not only manage their learning materials but also interact with their peers and schedule study sessions efficiently. This chapter explores the academic and technological landscape that has led to the development of this web application, highlighting the gaps in existing tools and the opportunities for improvement.

1.4 Relevance

The relevance of this research is rooted in the increasing demand for educational technology that addresses the evolving needs of students in modern academic environments. As more students move toward online and hybrid learning models, the necessity for integrated platforms that streamline the learning process becomes more evident. This web application offers a solution by combining document sharing, peer collaboration, and study session management into a single, user-friendly interface.

This application is highly relevant for current educational contexts, where students are often balancing multiple courses, assignments, and extracurricular activities. By providing a platform that enables students to access and organize learning materials, collaborate effectively with peers, and manage their study schedules, the app can significantly enhance productivity and academic performance.

The relevance extends beyond individual academic success; this research also contributes to the broader conversation around educational equity and technology access. By creating a tool that is easy to use and accessible, the application has the potential to benefit a wide range of students, including those from diverse backgrounds and those who may not have access to advanced learning management systems or collaboration tools.

Furthermore, the growing role of educational technology in shaping the future of education makes this research particularly significant. As the landscape of education continues to evolve, tools like this web application can pave the way for more efficient, inclusive, and collaborative learning environments, promoting a deeper connection between students and their academic communities.

Chapter 2

Related Work

2.1 Existing Solution

After careful research, some similar applications would include the following:

1. **Slack**[1] - Slack is a widely used team collaboration tool, particularly popular in professional and academic contexts. It enables users to create channels for specific topics or projects, making it suitable for organizing discussions and sharing resources. Additionally, Slack supports integrations with various applications and offers video call functionality, making it a versatile platform. However, Slack is designed primarily for professional teams and its interface may not specifically address the unique needs of student collaboration and study organization.

2. **Google Classroom**[2] - Google Classroom is widely used in educational settings for resource sharing and assignments. While it is more of an educational management tool, it integrates with Google Meet for video meetings and supports file sharing.

3. **Microsoft Teams**[3] - Teams is used extensively in academic and corporate settings for collaboration. It supports chats, file sharing, and video meetings, making it a good platform for group study environments.

4. **NutriBiochem Mobile Application**[4] - The NutriBiochem mobile application was developed as a learning resource for use in undergraduate biochemistry and nutrition education; specifically, the app was created for use in two courses in undergraduate Biochemistry and Nutrition at the University of Guelph Humber, although it was made freely available to the public through several different app stores and was promoted to students in several Biochemistry courses at the University of Guelph. While designed specifically to address the needs of these two specific courses, the content of the app (described below) is generalizable to most undergraduate courses that cover the fundamentals of human metabolism and nutrition. The goal of this project was to create a highly interactive, usable app that could be used as a mobile learning tool, and to study its use and pedagogical impact.

2.2 Gaps in Current Solution

Scientific articles can be found in different online database:

1. ACM Digital Library
2. Annual Reviews
3. IEEE Explore
4. JSTOR (social sciences, arts & humanities)
5. Science Direct
6. Springer Link
7. Wiley InterScience
8. SCOPUS
9. Google Scholar

Identify communities on the topic (specific workshops on national and international conferences)

Some of the articles are accessible through university IP.

Some other thesis on the topic an be consulted: `oatd.org`, `openthesis.org`, etc.

2.3 Proposed Solution

For each read paper try to note information, so you can use later:

- resource author;
- resource name (article title, book title, ...);
- subject: what is the paper aim;
- addressed problem ;
- methods/methodology used to solve the problem;
- used algorithms;
- test data (e.g. benchmarks, real cases).

For a git resource or an existing application:

- URL;
- application features;
- used technologies.

Chapter 3

Application Description

In this chapter present the application using code snippets, diagrams.

3.1 Architecture

The application uses a combination of Event Driven architecture and Request-Response architecture, making as much use as possible of Node JS's built in capabilities for handling multiple operations on a single thread using concurrency and asynchronous programming.

The application consists of multiple microservices. These microservices implement communication patterns like *Request-Response* and *Publish/Subscribe*. All microservice communication is done using message queues like RabbitMQ and BullMQ. While RabbitMQ offers great support for both Request-Response communication and Publish/Subscribe communication, I chose to use BullMQ for Publish/Subscribe. BullMQ uses Redis to queue messages and process them. By using Redis, BullMQ offers an easy way to schedule the times when a message is executed and implement retry policies. On the other hand, RabbitMQ is built upon AMQP (Advanced Message Queuing Protocol). This makes it possbile to pass messsages across an internal network and register handlers for these messages.

Since this application is heavily focused on microservices, here is a short description for each one of them:

- **Webserver Service** - Single HTTP server, tasked with serving the client application and providing HTTP routes for fetching data.

- **Authentication Service** - Service handling all authentication logic. This server is only accessible via RabbitMQ queues.

- **Core Service** - Service handling all reads and writes to the database. This server is accessbile via RabbitMQ queues.

- **Authorization Service** - Service handling authorization across the whole application. The application implements a RBAC (Role Based Access Control). Each request goes through this service before reaching the Core service. Only accessbile via RabbitMQ.

- **Uploader Service** - Service handling all file uploads. Since uploads are usually resource intensive tasks and can fail due to numerous reasons, this is a processor service using BullMQ.
- **Mail Service** - Service handling all emails sent. Sending emails is also a resource intensive task especially if sending to multiple users at the same tim. This is a processor service using BullMQ.
- **Websocket Server** - Server exposing only a websocket connection. This server is handling all of the chat features inside of the application, listening for notifications sent by clients, invoking the Core service for different writes (saves, updates) to the database, and dispatching a notification to the intended client.
- **Notification Server** - Server exposing a route for handling SSE (Server Sent Events). This server is the one handling in-app notifications.

Authentication Logic (e.g. Figure 3.1).
File upload Logic (e.g Figure 3.2).
Use Case Diagram (e.g Figure 3.3).

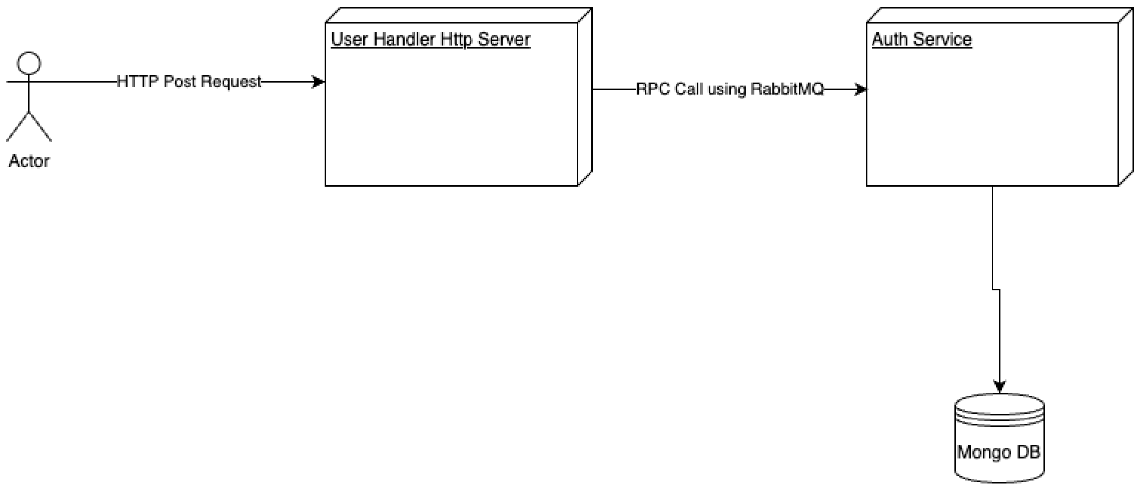

Figure 3.1: Authentication Logic Diagram

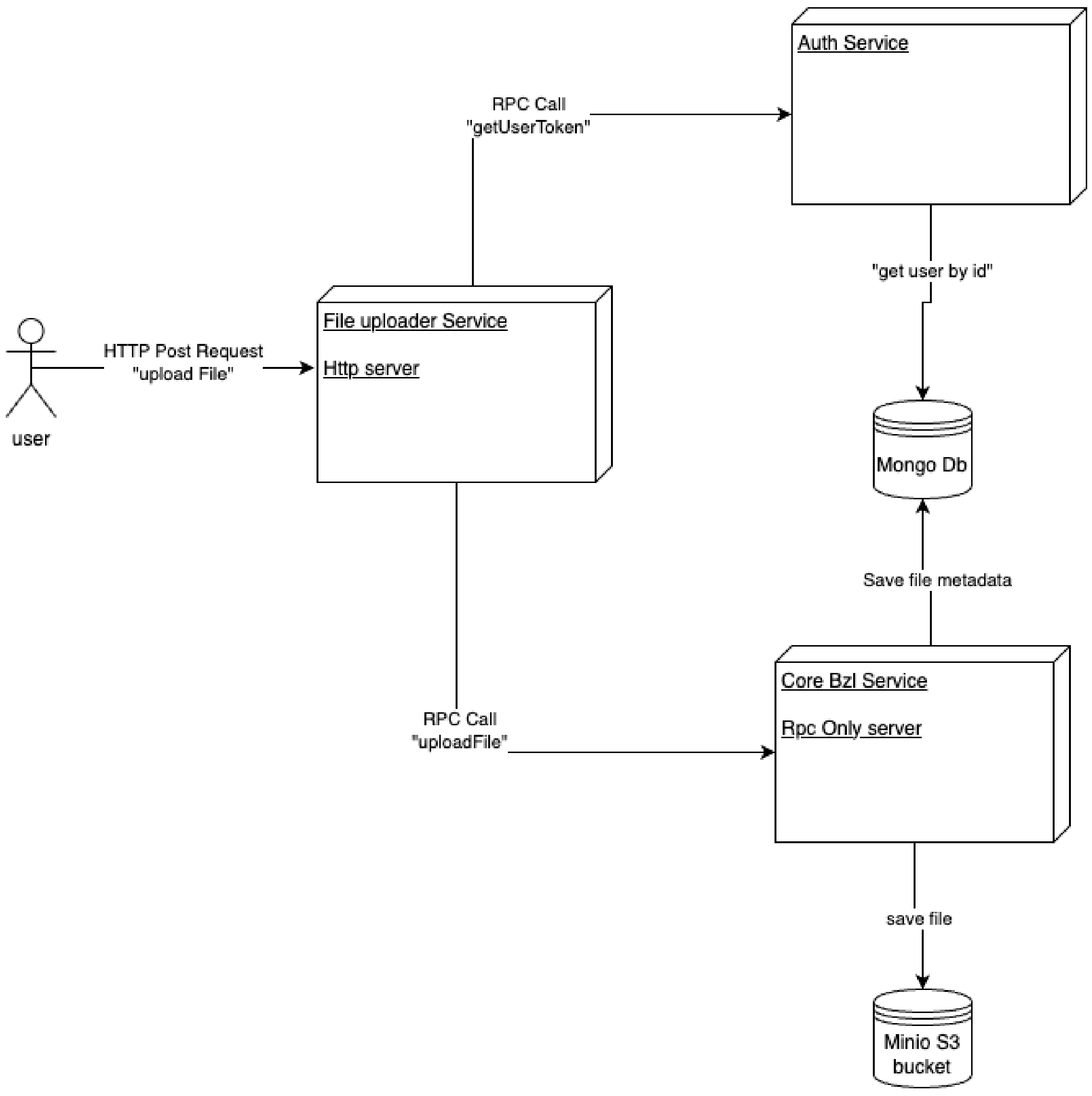

Figure 3.2: File Uploader Logic Diagram

3.2 Features

The main features of the application are: a steady authentication, a live-chat for allowing students to study in groups and a file-sharing system for easily sharing study notes.

3.2.1 Authentication

The most important part of any web based application is the authentication. While developing the authentication system for my application, I followed OAuth2 standards.

Registration: During registration, users submit their full name, email, password and a confirmation password. The system hashes the password with a strong algorithm (implemented using bcryptjs), creates a new user record marked as unverified, and generates a time-limited email verification token. The token is stored

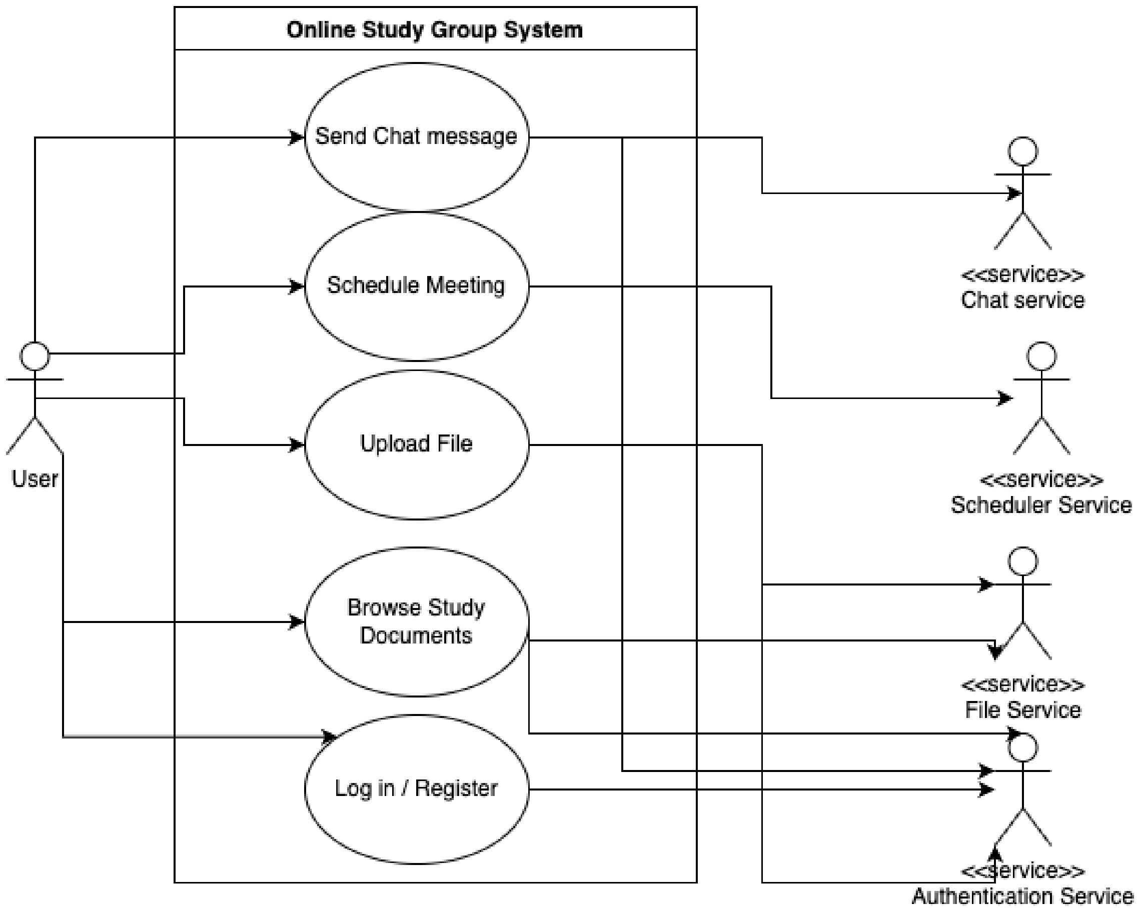

Figure 3.3: Use Case Diagram

in the database and emailed, prompting users to confirm their address.

Endpoint: /auth/register

Method: POST

Request Body: JSON Object containing fullname, email, password and confirmation password.

Response: JSON Object containing a success message if the request was successful

Verify Account: When users click the verification link, the system extracts the token and searches the database for a matching, unexpired record. If valid, it marks the user's account as verified, removes the token to prevent reuse, and returns a success response, enabling full access to login and other protected features.

Endpoint: /auth/verifyaccount

Method: GET

Query Parameters: verificationToken (string) -¿ verification token for certain account, recieved via email.

Response: JSON Object containing a success message if the request was successful

Login: On login, users submit credentials, which the system verifies against stored hashes. After confirming the account is verified, it generates a cryptographically secure session code, encrypts it, and stores it in the database under the user profile. The code is set in an HttpOnly, Secure cookie, establishing the authenticated session.

Endpoint: /auth/login

Method: POST

Request Body: JSON Object containing email, and password.

Response: JSON Object containing a success message if the request was successful.

Reset Password: When a password reset is requested, users provide their email address. The system generates a short-lived, cryptographically random reset token, stores it, and emails a reset link. Upon visiting the link and submitting a new password, the system validates the token, updates the password hash and deletes the token.

Endpoint: /auth/reset-password

Method: POST

Request Body: JSON Object containing reset token, password and confirmation password.

Response: JSON Object containing a success message if the request was successful

Logout: For logout, the client sends a request to invalidate the session. The server locates the encrypted session code in the database, removes it, and instructs the browser to clear the authentication cookie by setting it with a past expiration. The user is then unauthenticated for any subsequent requests.

Endpoint: /auth/logout

Method: POST

Response: JSON Object containing a success message if the request was successful.

Whoami: For Whoami, the client sends a request to get the user holding the current session. The server locates the encrypted session code in the database, and returns the user data.

Endpoint: /auth/whoami

Method: Get

Response: JSON Object containing userId, role and email of the logged in user if the request was successful.

3.2.2 Live-chat

3.3 Technology Stack

Since this is a web browser based application,it mainly consists of JavaScript/-TypeScript code on the client side. However, due to JavaScript‘s asynchronous nature, and the different microservices that must handle asynchronous operations, the server side code is also implemented in JavaScript/TypeScript.

3.3.1 Server Implementation

The server for ’Cloud Class’ contains all of the logic at the backbone of the application, offering a solid foundation for the client‘s app functionalities.

- **Programming Language:** *TypeScript* - TypeScript is a statically typed superset of JavaScript that compiles down to plain JavaScript. It introduces optional type annotations, interfaces, and advanced tooling for improved code quality, maintainability, and developer productivity, enabling safer large-scale application development.

- **Runtime Environment:** *Node.js* - Node.js is a cross-platform, event-driven JavaScript runtime built on Chrome's V8 engine. It uses non-blocking I/O and an event loop to efficiently handle concurrent connections, making it ideal for scalable network applications, microservices, and real-time servers.

- **Framework:** *Nest.js* - NestJS is a progressive Node.js framework built with TypeScript, leveraging decorators and dependency injection to structure applications into modular, testable components. It abstracts common server-side patterns, integrates seamlessly with Express or Fastify, and supports GraphQL, WebSockets, and microservices out of the box.

- **Database:** *MongoDB* - MongoDB is a document-oriented NoSQL database that stores data in flexible, JSON-like BSON documents. It offers horizontal scalability through sharding, built-in replication for high availability, and a powerful query language. MongoDB suits dynamic schemas, real-time analytics, and modern web or mobile applications.

- **Storage:** *MinIO S3* -MinIO is a high-performance, Kubernetes-native object storage server compatible with the Amazon S3 API. It provides erasure coding, bitrot protection, and encryption for secure, scalable storage. MinIO excels in private cloud or edge deployments, supporting multi-tenancy and lightweight resource footprints.

- **Real Time Communication:** *Socket.IO and Server Sent Events (SSE)* - Socket.IO enables real-time, bidirectional communication between clients and servers over WebSockets and fallback transports, ideal for chat, gaming, or live updates. Server-Sent Events (SSE) provide unidirectional, low-overhead streams from server to browser. Both facilitate reactive interfaces and event-driven architectures.

- **Logging:** *Elastic Search* - Elasticsearch is a distributed, RESTful search and analytics engine built on Apache Lucene. It indexes documents for lightning-fast full-text, structured, and geo-queries. Elasticsearch scales horizontally, supports real-time data ingestion, and powers features like autocomplete, logging pipelines, and business intelligence dashboards.

- **Testing:** *Jest* - Jest is a delightful JavaScript testing framework maintained by Facebook. It offers zero-configuration setup, snapshot testing, and built-in mocking utilities. Jest runs tests in parallel for speed and provides rich watch-mode feedback, making test-driven development smooth for React, Node.js, and beyond.

- **Message Queues:** *RabbitMQ and BullMQ* - RabbitMQ is an open-source message broker implementing AMQP. It offers flexible routing, message acknowledgments, clustering, and high availability. Through plugins and management UI, you can monitor queues and extend functionality. RabbitMQ enables reliable, asynchronous communication between distributed applications and microservices. BullMQ is a Node.js library for robust background job processing using Redis. With a TypeScript-first API, it supports job queues, workers, rate limiting, repeatable jobs, and atomic operations. BullMQ ensures reliable task scheduling for use cases like email delivery, data pipelines, and notification dispatch.

3.3.2 Client Implementation

The client application for 'Cloud Class' offers a easy-to-use interface, allowing users to collaborate easily with each other.

- **HTML:** - HTML (Hyper Text Markup Language) is the standard markup language for creating web pages. It structures content using elements like headings, paragraphs, links, and media embeds. Semantic tags improve accessibility and SEO, while forms and tables enable user input and data presentation, forming the backbone of every website.

- **CSS:** - CSS (Cascading Style Sheets) styles and layouts HTML content, controlling colors, typography, spacing, and responsive behavior. With selectors, properties, and media queries, it allows designs to adapt across devices. Modern features like Flexbox, Grid, and custom properties enable complex, maintainable, and dynamic user interfaces with minimal code.

- **React:** - React is a JavaScript library for building component-based user interfaces. It uses a virtual DOM for efficient updates and a declarative syntax with JSX. React's ecosystem includes hooks for state and side effects, and tools like React Router and Redux, making it ideal for complex, interactive web apps.

- **Socket.IO:** - React is a JavaScript library for building component-based user interfaces. It uses a virtual DOM for efficient updates and a declarative syntax with JSX. React's ecosystem includes hooks for state and side effects, and tools like React Router and Redux, making it ideal for complex, interactive web apps.

3.4 References

The bibliography has to be referenced in thesis content using cite (e.g. [5]).

3.5 Interface

Present App interface using Screenshots?

Each figure has to have a caption that is a suggestive description of what the picture represents (e.g. Figure 3.4).

Figure 3.4: FMI logo scaled at 25% of text width

3.6 Algorithm pseudo-code

Present pseudo-code and actual code for a feature.

Pseudo-code is a formal way to describe an algorithm, is more clear than a textual description ore code (e.g. Algorithn 1).

Algorithm 1 An algorithm with caption

```
Require: $n \geq 0$
Ensure: $y = x^n$
 1: $y \leftarrow 1$
 2: $X \leftarrow x$
 3: $N \leftarrow n$
 4: while $N \neq 0$ do
 5:     if $N$ is even then
 6:         $X \leftarrow X \times X$
 7:         $N \leftarrow \frac{N}{2}$                    ▷ This is a comment
 8:     else if $N$ is odd then
 9:         $y \leftarrow y \times X$
10:         $N \leftarrow N - 1$
11:     end if
12: end while
```

3.7 Adding code

If it necessary to add some code from the application, do not use print-screens, use listing (e.g. listing 3.1).

Listing 3.1: Un exemplu de cod python

```python
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

3.8 Database Structure

Present Strucutres for all database documents used in the application, using code snippets, diagrams, photos

Bibliography

[1] "What is Slack?" [Online]. Available: https://slack.com/help/articles/115004071768-What-is-Slack

[2] "What is Google Classroom." [Online]. Available: https://edu.google.com/workspace-for-education/products/classroom/

[3] "What is Microsoft Teams." [Online]. Available: https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29

[4] "NutriBiochem Application." [Online]. Available: https://iubmb.onlinelibrary.wiley.com/doi/full/10.1002/bmb.20771

[5] M. M. Bersani, M. Erascu, S. Ghilardi, F. Marconi, and M. Rossi, "Formal Verification of Data-Intensive Applications through Model Checking Modulo Theories."