NEXTECH
INNOVATIONS

LECTORO DESIGN REPORT

Facial Recognition Attendance System

Dr. Fenghui Ren

Presented By
MATTHEW HEALEY
IDRIES EAGLE-MASUAK
SAMUEL FITZPATRICK-FULLER
JAMES KIM
GIA BACH NHU
DINH QUOC HUY NGUYEN

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The design report for the Lectoro Automatic Attendance System demonstrates a meticulously crafted effort to building an effective solution specific to UOW's educational environment for lecture attendance. This comprehensive document describes our effort to conceptualise, analyse, and create a cutting-edge system that will significantly improve attendance management at UOW.

Design Methodology:
The design methodology prioritizes scalability, maintainability, and user-centricity under the SCRUM managment framework. Through iterative prototyping, usability testing, and continuous collaboration under an agile approach, the design team ensures alignment with stakeholder expectations and industry best practices.

Requirement's Analysis:
The foundation of the objectives is a thorough requirements analysis that takes into account the demands of stakeholders, system functionality, and technological factors. User-friendly interfaces, strong data protection measures, and seamless integration are essential for Lectoro's success at UOW.

Tools:
A suite of development and deployment tools empowers the design process, facilitating efficient communication, version control, and system deployment. Notable tools include Visual Studio Code for development, Git and GitHub for version control, and Amazon Web Services for deployment.

Back-end Design:
The back-end architecture leverages Java and Spring Boot for high performance and scalability and Integration with MongoDB ensures flexible data management.

Development Environment:
Python will serve as the primary language for facial recognition tasks, while Java powers the back-end infrastructure. Integrated Development Environments (IDEs) such as Visual Studio Code and IntelliJ streamline coding workflows, while Git and GitHub ensure version control.

Deployment Environment:
Amazon Web Services (AWS) provides a scalable and secure hosting platform for the live deployment of Lectoro. Amazon EC2 and S3 facilitate compute capacity and static as-set storage, ensuring reliable system performance and data management.

Front-end Design:
The front-end adopts a dynamic HTML approach complemented by CSS and JavaScript, prioritizing user experience and interface consistency. Tailwind CSS and Flowbite enhance rapid development while maintaining code reusability and scalability and uses RESTful API's to communicate with the back-end.

Stakeholder and System Scenarios:
Stakeholder analysis highlights the roles, responsibilities, challenges, and strategies of key stakeholders, ensuring comprehensive engagement and alignment with project objectives. Detailed system scenarios elucidate the operational flows, preconditions, and contingencies of Lectoro, offering a comprehensive blueprint for system functionality in real-world settings.

INTRODUCTION

We here at NexTech Innovations are pleased to provide our comprehensive design report for Lectoro, a ground-breaking face recognition attendance system set to transform the landscape of attendance recording and management at The University of Wollongong (UOW). Lectoro is created for the respected UOW, specifically it's lecture environments and cooperation with sponsor Dr. Fenghui Ren, and it represents the culmination of inventive thought, meticulous planning, and commitment to quality in educational technology.

This design report reflects the extensive research that has been undertaken and the journey that will be taken to conceptualise, design, and create Lectoro, with the overriding objective of creat-ing a solution that improves the lecture experience for all stakeholders. At the core of NexTech Innovations and our efforts is a dedication to using cutting-edge technology and industry best practices to meet the changing requirements and problems that UOW and other academic insti-tutions confront in today's digital age.

Lectoro's design process is characterised by a methodical and disciplined approach, led by a common goal to produce a solution that not only fits the functional needs of attendance record-ing and monitoring but also aims to surpass in areas of usability, dependability, and flexibility. From the early stages of ideation and conceptualisation to the final phases of product develop-ment and deployment, we have methodically planned every step of the design process and plan to execute our vision successfully.

This design report provides a thorough examination of the many aspects of Lectoro's design and development plan. We dig into the complexities of our chosen design techniques, offering in-sights and reasons into the frameworks, processes, and principles that influenced our decision-making in our Front-end and Back-end design. In addition, we provide extensive details of the development and deployment environments, the tools and technologies that will be used, and the complete requirements analysis performed to establish the project's scope.

In addition to stakeholder feedback, we highlight the thorough prioritisation approach used to de-termine and assign importance to objectives based on criteria such as feasibility, effect, and user value. We also look at the selection of stretch goals aimed at improving the system's capabilities, as well as the reasoning behind their inclusion or absence from the prioritised needs list.

We encourage you to learn about Lectoro's design process and celebrate the spirit of develop-ment, co-operation, and quality that distinguishes this effort at NexTech Innovations. We hope to showcase our upcoming journey of developing a cutting-edge solution that promises to improve attendance recording and monitoring and raise the educational landscape at the University of Wollongong.

DESIGN METHODOLOGY

The design processes that will be used in the development of the Lectoro is critical to its success, emphasising an agile and user-centric approach. In this part, we look at the whole design process, which is showcased by the concepts of collaboration, iterative refinement, and stakeholder involvement. By embracing SCRUM approaches, following to design principles, including user-centered practices, and encouraging effective collaboration and communication, we want to create a strong, intuitive, and scalable system that satisfies the demands of our team at NexTech, users and stakeholders.

OVERVIEW OF DESIGN PROCESS

Lectoro is designed around using the SCRUM technique, which is a prominent agile framework that emphasises iterative development and cross-functional interaction. Following SCRUM, the design process is divided into iterative sprints, which run from two to four weeks. During each sprint, both front-end and back-end teams works on completing a set of prioritised tasks that are established by stakeholder needs and project objectives. The key steps of the design process are:

Design:
To help visualise the user interface and system architecture, our team will produce wireframes, mockups, and prototypes. Usability, accessibility, and scalability are concepts that govern our design decisions.

Implementation:
Our team will translate the design principles into practical code using the best practices and coding standards ultimately to meet project objectives successfully. Continuous integration and version control technologies help to assure code quality and team cooperation.

Testing:
Our team will thoroughly test to detect and address any bugs or issues in the system and its processes. Automated testing frameworks will be used to improve testing efficiency and assure consistent performance across several settings.

Deployment:
Once Lectoro has passed testing and fulfils the acceptance requirements, it is moved to a production environment for active usage. Continuous monitoring and feedback loops guarantee continuous optimisation and improvement.

A Gantt chart is used to visually represent the timing and dependence of each phase, allowing for better project management and progress tracking. This iterative method allows the team to react to changing requirements and stakeholder feedback, ensuring that the end product satisfies user needs and expectations while sticking to a structured schedule.

DESIGN PRINCIPLES

The design decisions for both the front end and back end of the Lectoro system are guided by several key principles:

Front-end Design Principles:
To improve the user experience, we emphasise simplicity, intuitiveness, and accessibility. Clean design, clean typography, and straightforward navigation structures prioritise usability. Consistency in design aspects is achieved by a visual style guide that highlights the designs of colours, fonts, inputs, tables, and other features, as well as adherence to accessibility standards, which ensures inclusiveness for all users.

Back-end Design Principles:
Architectural considerations are based on efficiency, scalability, and maintainability. A monolithic design with Spring Boot is chosen for its simplicity and ease of deployment, as well as the flexibility to scale in the future via microservices. Database design prioritises flexibility and data integrity, resulting in easy data storage and retrieval.

USER-CENTERED DESIGN APPROACH

During the testing phase, usability testing is incorporated in the design process to ensure that Lectoro satisfies the needs and expectations of its users on the front-end side, specifically Lectoro's intended management tools. Throughout the design phase, the team will be collecting input through our team and sponsor where this data will guide our decisions on how information will be displayed for report creation, guaranteeing clarity, efficiency, and user pleasure. Iterative refinement based on customer feedback guarantees that the product improves and aligns with consumer demands over time.

ITERATIVE DESIGN PROCESS

The design is developed through iterative cycles of prototyping, testing, and refining. Low-fidelity wireframes and mockups are first generated to help visualise the system's layout and functioning. These prototypes are then tested with users to collect feedback and identify areas for development on both the front-end and backend. Based on user feedback, the design is continually modified, with new versions created and tested until the required level of usability and functionality is attained. This iterative method provides for greater flexibility and adaptation to changing requirements and user needs, ultimately resulting in a more robust and user-friendly end product.

COLLABORATION AND COMMUNICATION

The success of Lectoro's design process depends on effective cooperation among team mem-bers as well as communication with our sponsor and potentially stakeholders. Regular meetings promote cooperation by ensuring that team members are aligned with our objectives and pro-gress is transparent. Sponsor feedback is sought at critical phases of the development process, allowing for the early creation of requirements and possible concerns to be discovered as soon as possible. Open lines of communication within the team and with our sponsor promote a col-laborative and iterative approach to design, resulting in ongoing improvement and innovation.

Ultimately, the design methodology used to develop the Lectoro Automatic Attendance System represents a comprehensive approach, combining agile SCRUM methodologies, user-centered design principles, iterative refinement cycles, and effective collaboration and communication strategies. By following these ideas and methods, we want to design a system that not only achieves functional needs but also prioritises user happiness, scalability, and stakeholder participation. We seek to guarantee that the Lectoro system matures into a robust and intuitive solution that successfully answers the demands of educators, administrators, and students by iterating, refining, and soliciting input from stakeholders.

REQUIREMENT’S ANALYSIS

The Requirements Analysis of Lectoro's development is critical for determining the system's scope and functions. In this part, we describe the key features, additional requirements, and stretch goals that will define the system's design and development. We prioritise requirements according to feasibility, impact, and user value, ensuring that the final product meets stakeholder demands and project objectives. Join us as we delve into the complexities of this phase, with the goal of developing a breakthrough system that will revolutionise academic attendance management for UOW.

Facial Recognition Attendance System:
This includes the development of creating a sophisticated facial recognition system that can reliably identify and scan the faces of students attending lectures at the University of Wollongong (UOW). The system should be built to work efficiently with current UOW lecture room infrastructure and to accurately record student attendance in real time.

Add, Edit, Delete and Assign Subjects:
Administrators should be able to control subjects offering large scale lectures at UOW by adding, modifying, removing, and assigning them to specific lecturers. This feature provides the flexibility to adapt changes in course offerings and lecturer assignments as needed.

Manage Student Enrolment:
Administrators should be able to track and manage enrolment of students for each subject, as well as analyse and update the registered student list. This functionality simplifies the process of handling student data and guarantees proper attendance monitoring.

Access Comprehensive Attendance Data:
Administrators should have access to detailed attendance data for all subjects, allowing them to track trends, discover patterns, and produce reports for examination. This tool helps administrators to make educated judgements and take appropriate steps towards improving attendance rates and student engagement.

Enhanced Data Displays:
The system should offer attendance data in an understandable and visually appealing format, with colour-coded indications representing various attendance statuses such as arrival, late arrival, late departure, and absence. Our inspiration for enhanced data displays came from modern data visualisation tools like Tableau and Power BI, which offer visually appealing (*Power BI - Data Visualization | Microsoft Power Platform* 2023) and customizable (*Tableau Desktop* 2024) dashboards with color-coded indicators for easy interpretation of data. This improved data display for Lectoro increases readability and understanding, allowing users to rapidly comprehend attendance data briefly.

Reporting Functionalities:
The system should offer comprehensive reporting capabilities to provide detailed attendance records, analyse trends, and extract insights for decision-making. We researched reporting functionalities in learning management systems (LMS) like Canvas by Instructure (*Canvas by Instructure | World’s #1 Teaching and Learning Software* 2021). These systems provide comprehensive reporting capabilities allowing administrators and instructors to generate detailed attendance records, analyze trends, and extract insights for decision-making and planning. In Lectoro, reporting tools will allow administrators and lecturers to monitor attendance indicators over time, discover patterns, and make decisions to increase student engagement and academic performance.

STRETCH GOALS

Automated Attendance Notification:
Implement a notification alert system that will automatically inform students if they are not identified and therefore not accounted for by the face recognition system during lectures. These reminders can be delivered via email, SMS, or push notifications to remind students to check in and confirm their attendance, lowering the possibility of missing records and encouraging accountability. The student can have further control by confirming their status in terms of options they're given in the notification. If a student is not attending, in their notification they can state they're not here and optionally attach the reason why with a message. If a student is here, but not accounted for by the facial recognition system, then the student can select an option to access a QR code scanner.

QR Code Attendance Confirmation:
Allow students to manually confirm their attendance by scanning a QR code particular to the lecture room after receiving an absence notification. Our influence comes from this extra capability improves the accuracy and completeness of attendance monitoring, particularly in situations when facial recognition is not feasible, efficient or an error occurs. By providing numerous means of attendance confirmation, the system guarantees flexibility and dependability in obtaining student attendance data without the need of stepping back to traditional attendance tracking.

SPONSOR INPUT

Feedback from our sponsor Dr. Fenghui Ren has been instrumental in shaping the project requirements for Lectoro. As the primary key decision-maker for our objectives and scope, Dr. Fenghui Ren has provided invaluable insights and guidance throughout the project development process. Collaborative discussions with him, along with consultations aimed at understanding the perspectives of university administrators, lecturers, and students, have been paramount in forming the development of key features and functionalities.

In particular, Dr. Ren's input has been pivotal in identifying the need for comprehensive administrative and lecture attendance management capabilities within the system. By understanding the specific requirements and challenges faced by administrators and lecturers in managing course materials, providing students with learning, and attendance, we have been able to tailor the system to meet their needs effectively. Additionally, Dr. Ren's feedback has guided the implementation of enhanced data display features, ensuring that attendance information is presented in a clear, user-friendly manner for easy interpretation and analysis.

Moreover, Dr. Ren's active involvement in project discussions has led to the identification of stretch goals aimed at further enhancing the system's functionality. Specifically, the inclusion of automated attendance notifications to account for potential anomalies and QR code attendance confirmation have been driven to create a vision of a robust and versatile attendance tracking solution at UOW. These stretch goals align closely with user needs and expectations, as identified through ongoing communication and collaboration with stakeholders.

In summary, Dr. Fenghui Ren's engagement and feedback have been critical in shaping the project requirements and ensuring alignment with the needs and expectations of all stakeholders that are to be involved. By leveraging his insights and expertise, we are confident in our ability to deliver Lectoro as a comprehensive facial recognition attendance system that meets the diverse needs of the University of Wollongong.

OBJECTIVE PRIORITISATION

Requirements were meticulously prioritised based on a comprehensive assessment of several key factors, including feasibility, impact, and user value. This prioritisation process aimed to en-sure that Lectoro and its objectives remained focused on delivering maximum value to stake-holders while staying within practical constraints.

The core objectives of the project revolved around implementing essential features that directly addressed the primary need for an efficient attendance tracking system. As such, critical functionalities such as the facial recognition attendance system, admin and lecturer management capabilities, and basic data display were accorded the highest priority. These foundational features were deemed indispensable for achieving the project's overarching goal of automating attendance tracking and streamlining administrative management tasks.

In addition to the core functionalities, certain additional requirements were identified to enhance the overall user experience and facilitate comprehensive data analysis. These included features such as enhanced data display options and robust reporting functionalities. Prioritising these enhancements is essential to ensure that Lectoro not only met the basic re-quirements, but also provided Administrators and Lecturers with intuitive tools for interpreting at-tendance data and deriving actionable insights.

Furthermore, stretch goals were considered as potential objectives for further enhancing the sys-tem's capabilities. Stretch goals like an automated attendance notifications for student anomalies and QR code attendance confirmation were identified as ambitious yet valuable additions that have the potential to significantly improve attendance tracking accuracy and enhance user en-gagement. However, it’s acknowledged that some stretch goals, such as Lectoro itself and it’s attached data synchronised within the university's existing Moodle student information system, would pose significant technical challenges and integration complexities to fit inside our time constraints. Despite their potential benefits, these stretch goals were deemed infeasible within the project's scope and timeline.

Ultimately, the prioritisation process involved a careful balancing act, weighing the feasibility, impact, and resource implications of each requirement against the project's objectives and constraints. While certain stretch goals were excluded from the prioritised requirements list due to their technical challenges or resource constraints, their potential value is duly acknowledged. This iterative prioritisation approach ensured that the project remained focused on delivering tangible benefits to stakeholders while maintaining a realistic and achievable scope.

TARGET MARKET

The market analysis reveals a significant amount of focus on HR and employee settings among the products investigated, offering insightful information about the varied landscape of attendance management systems. Timeero, Quickbooks Time, and Buddy Punch are just a few examples we researched of the carefully designed products that are meant to meet the complicated needs of companies in a variety of industries where workforce optimisation, effective time tracking, and attendance management are critical. Numerous features catered to the needs of companies and HR departments are available in these solutions, such as geofencing capabilities (*Jibble* 2024), real-time notifications (Samson Kiarie 2024), and extensive reporting tools (*Facial Recognition Feature | Software Uses & Features | Buddy Punch* 2024) designed to boost efficiency and guarantee adherence to labour laws.

In contrast to the HR-centric focus of the products reviewed, our proposed solution is strategically positioned to address the unique challenges faced by educational institutions, particularly in large lecture settings with high student enrolments specifically for the University of Wollongong lecture environments. Unlike traditional HR-focused attendance management systems, our product is solely intended to simplify the attendance marking process for students attending lectures, workshops, or other academic activities. This targeted approach ensures that our solution is fine-tuned to meet the unique needs of the educational sector, where accurate attendance tracking is critical in monitoring student engagement, participation, and academic progress.

Furthermore, while the products examined in the market analysis include a variety of features geared towards workforce management, such as employee time tracking and performance analytics, our product prioritises simplicity, accuracy, and efficiency in recording student attendance. By focusing on academics, we can tailor our solution to integrate seamlessly with accommodating the unique needs of large class sizes and provide educators with actionable insights for improving the overall learning experience.

In conclusion, while the products examined in the market analysis serve the HR and employee management markets, our proposed solution is strategically aligned with the unique needs of educational institutions, particularly in large lecture settings for the University of Wollongong. By focusing solely on the academic sector, we hope to provide a comprehensive attendance management solution that not only improves administrative efficiency but also promotes student engagement, accountability, and academic success in UOW's dynamic lecture learning environments.

TOOLS

The success of every software development project is dependent not only on the team's compe-tence, but also on the tools and technologies used throughout the development process. In this part, we will briefly look at the tools that our team will be using during the development, collabo-ration, testing, and deployment of Lectoro. Each tool plays a critical part in the project's success. By carefully selecting and integrating these technologies into our workflow, we want to increase productivity, improve collaboration, expedite development processes, and eventually produce a robust and dependable solution to fulfil our stakeholders' expectations.

DEVELOPMENT TOOLS

Integrated Development Environment (IDEs):
The team will mostly used Visual Studio Code (VSCode) (Microsoft 2021) for coding and debugging. Its lightweight yet strong architecture (Microsoft 2021), along with a large library of extensions (Microsoft 2021), made it an effective development environment for both frontend and backend activities. Another IDE being IntelliJ, With its powerful code editing features, IntelliJ IDEA offers refactoring tools, intelligent code completion, and on-the-fly code analysis (JetBrains 2024).

Version Control:
Git and GitHub will be critical components of the development process, allowing for version control and team collaboration. GitHub's repository management capabilities (*Repositories documentation - GitHub Docs* 2024) and powerful branching procedures protected code integrity and facilitated collaboration.

DEPLOYMENT TOOLS

Hosting Platform:
The live system will be deployed and managed using Amazon Web Services (AWS). AWS's scalable infrastructure and comprehensive portfolio of services (*Cloud Computing Services - Amazon Web Services (AWS)* 2024), including Amazon EC2 for computing capacity (*AWS Amazon EC2 2023 - Compute Innovation Talk* 2023) and Amazon S3 for static asset storage (*Amazon S3 - Cloud Object Storage - AWS* 2024), provides a dependable and scalable hosting environment for the facial recognition system.

The tools we will be using to help development, and ease into the deployment stages of the Lectoro Automatic Attendance System are critical to our project's success. Visual Studio Code (VSCode) and IntelliJ IDEA are effective IDEs for coding and debugging, while Git and GitHub provided version control and facilitated collaboration. Integrating these technologies for Lectoro will increase efficiency, improve cooperation, and generate a comprehensive solution that will meet stakeholder expectations.

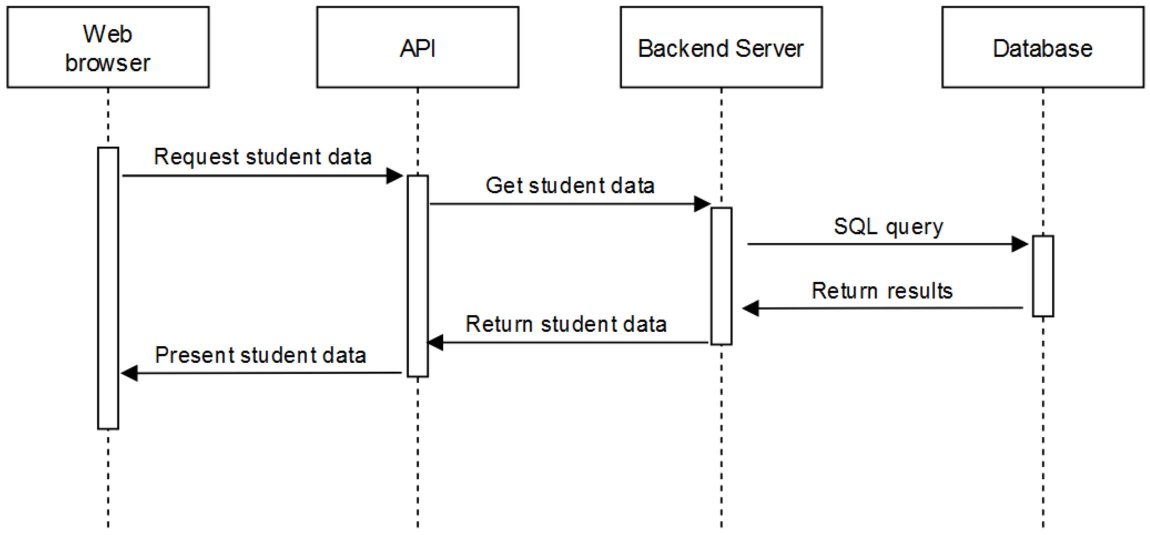
BACK-END DESIGN

Lectoro's powerful and efficient back-end architecture will serve as the foundation for smooth connections, data processing, and system functionality. This section provides a complete description of the back-end architecture, with an emphasis on major areas such as database interface, architectural decisions, and server-side technology.

DATABASE COMMUNICATION

For communication between back-end to the database, the Spring Boot backend application will interact with MongoDB using a Java driver library (*The Backend With Java and Spring Boot* 2020). This library provides functionalities to connect to the database, perform CRUD operations on documents, and execute queries. Popular choices for Java drivers include the official MongoDB Java Driver and Spring Data MongoDB.

Diagram 1. Sequence Diagram



The above diagram is used as an example to showcase the processes for if a Lecturer wants to view a student's information such as their first and last name:

The lecturer uses Lectoro's web app to request student information. The frontend sends a request to the backend via an API which queries the student database, returning the student's information back to the web browser via the API.

BACK-END ARCHITECTURE

Our selected back-end architecture for Lectoro follows a Spring Boot monolithic approach, with a single application including all features. This simplified design streamlines the development, implementation, and maintenance procedures (*Spring Modulith* 2024), providing a coherent and integrated solution for the automated attendance system. This means a single application will handle all functionalities, including:

Receiving Video Feeds or Images from Cameras:
The installed cameras in the lecture rooms will be able to send pictures or video feeds to the back-end system. The streams will be sent to the back end so that facial recognition may be processed.

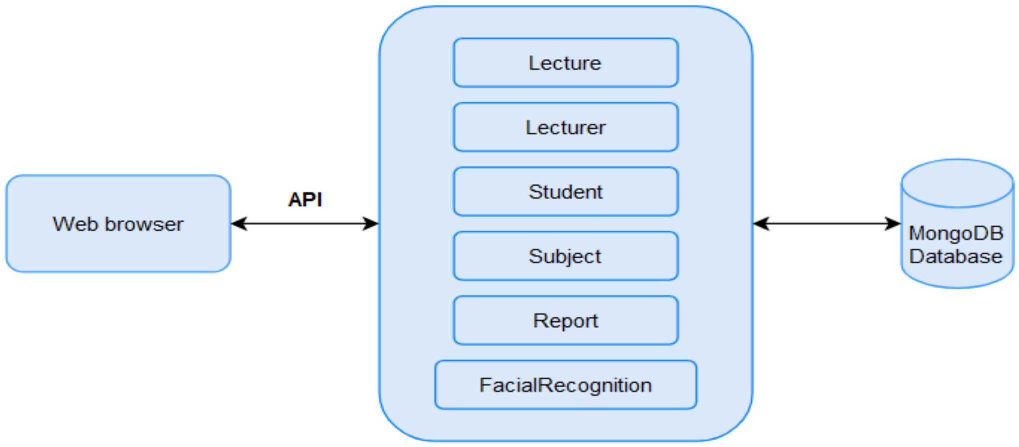
Triggering Facial Recognition using Python Libraries:
The back-end system will use Python libraries to initiate facial recognition upon receiving video feeds or photos. After identifying faces in the photos through analysis and comparison with current student records, these libraries will record attendance appropriately.

Storing Attendance Records and Student Data in MongoDB:
MongoDB will house attendance records and related student data. This data will be stored and retrieved by the back end

Handling User Interactions:
All user interactions, including those from lecturers, administrators, and students, will be handled by the back-end system. Users will be authenticated, their requests will be processed, and suitable responses will be sent depending on their roles and permissions inside the system.

Generating Attendance Reports and Trends Analysis:
The back end will create detailed reports and perform trend analysis using the attendance records that have been maintained. Administrators and teachers will be able to make well-informed judgements and take appropriate action to raise student engagement and attendance rates by using the insights these reports will offer about attendance patterns, trends, and statistics.

Diagram 2. High-level Architectural Diagram



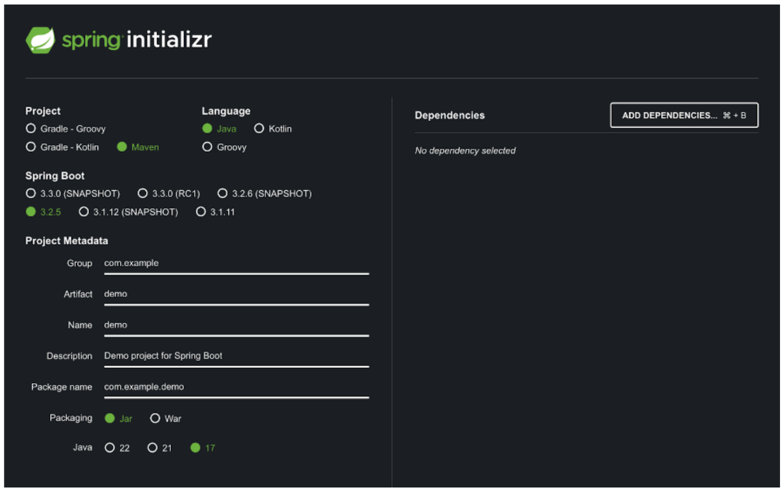
Using a single API server to house all system components, the monolithic architecture eliminates the need for several API servers and allows for smooth communication between back-end components. Because all of the code is centrally located within one environment, this architectural approach simplifies the development process. Developers may effectively manage and maintain the back end by minimising complexity and fostering code coherence when all components are colocated.

The monolithic architecture's ease of deployment is one of its main benefits. When all features are contained in one jar file, the deployment procedure is simplified and made less complicated (*Introducing Spring Modulith* 2022). This makes the system's maintenance and management lifecycle simpler by lowering deployment overhead and guaranteeing consistency across environments.

The monolithic architecture presents an acceptable option for Lectoro's back end, offering a solid basis for effective communication, streamlined development, and simple deployment.

SPRING BOOT

Spring Boot provides several advantages, such as boilerplate code-driven development acceleration, easier configuration, and future scalability through microservices architecture (*Spring | Why Spring* 2024). The project's development and maintenance procedures are also improved by its large community and documentation, which offer plenty of resources and assistance for our team at NexTech Innovations to review.



spring initializr

Project
Gradle - Groovy | Gradle - Kotlin | [✓ Maven]

Language
[✓ Java] | Kotlin | Groovy

Spring Boot
3.3.0 (SNAPSHOT) | 3.3.0 (RC1) | 3.2.6 (SNAPSHOT) | [✓ 3.2.5] | 3.1.12 (SNAPSHOT) | 3.1.11

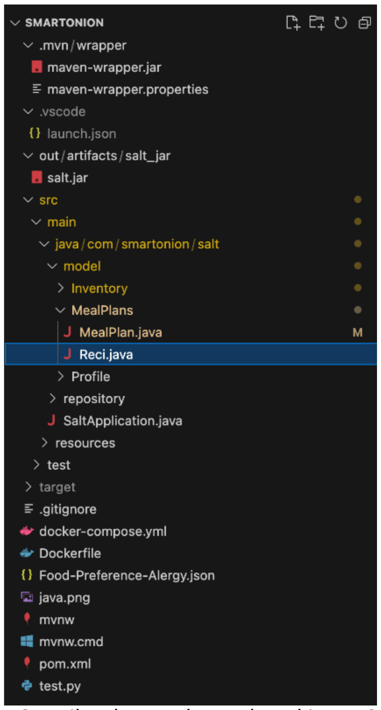
Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	22 21 [✓ 17]

Dependencies

No dependency selected

Figure 1: Spring Boots configuration settings



```
SMARTONION
  .mvn / wrapper
    maven-wrapper.jar
    maven-wrapper.properties
  .vscode
    launch.json
  out / artifacts / salt_jar
    salt.jar
  src
    main
      java / com / smartonion / salt
        model
          Inventory
          MealPlans
            MealPlan.java                M
            Reci.java
          Profile
        repository
        SaltApplication.java
      resources
    test
  target
  .gitignore
  docker-compose.yml
  Dockerfile
  Food-Preference-Alergy.json
  java.png
  mvnw
  mvnw.cmd
  pom.xml
  test.py
```

Figure 2: Boilerplate code produced into VSCode IDE.

While Spring Boot's modular architecture enables future scaling through microservices (*Spring | Microservices* 2024), it initially provides a monolithic architecture for ease of use and quick development. Our project in the future may potentially achieve better flexibility and scalability by dividing the application into smaller, independent services, each responsible for distinct functionalities. Our team may deploy and scale individual components independently with microservices architecture, which optimises resource allocation (*Spring | Microservices* 2024) and accommodates different workloads. The flexibility of Spring Boot makes it possible to go from a monolithic to a microservices design, opening up possibilities for long-term scalability and responsiveness to changing business requirements.

Spring Boot benefits from a vibrant and extensive community of developers, architects, and enthusiasts. This large community fosters collaboration, knowledge-sharing, and continuous improvement of the framework which can be found at https://spring.io/projects/spring-boot. Overall, Spring Boot's robust community and documentation contribute to its popularity and usability, making it a reliable choice for Lectoro.

When integrating Python with Java Spring Boot, there are two main approaches to consider, each with its own advantages and disadvantages:

ProcessBuilder Class:
One approach includes using Java's ProcessBuilder class to run Python scripts as external processes (MyExamCloud 2023). This method is relatively simple and does not necessitate a complex setup. However, it may not be suitable for applications that include complicated interactions between Python and Java components. While it is appropriate for simple use cases, it may lack the strength and efficiency required for more complicated functionalities.

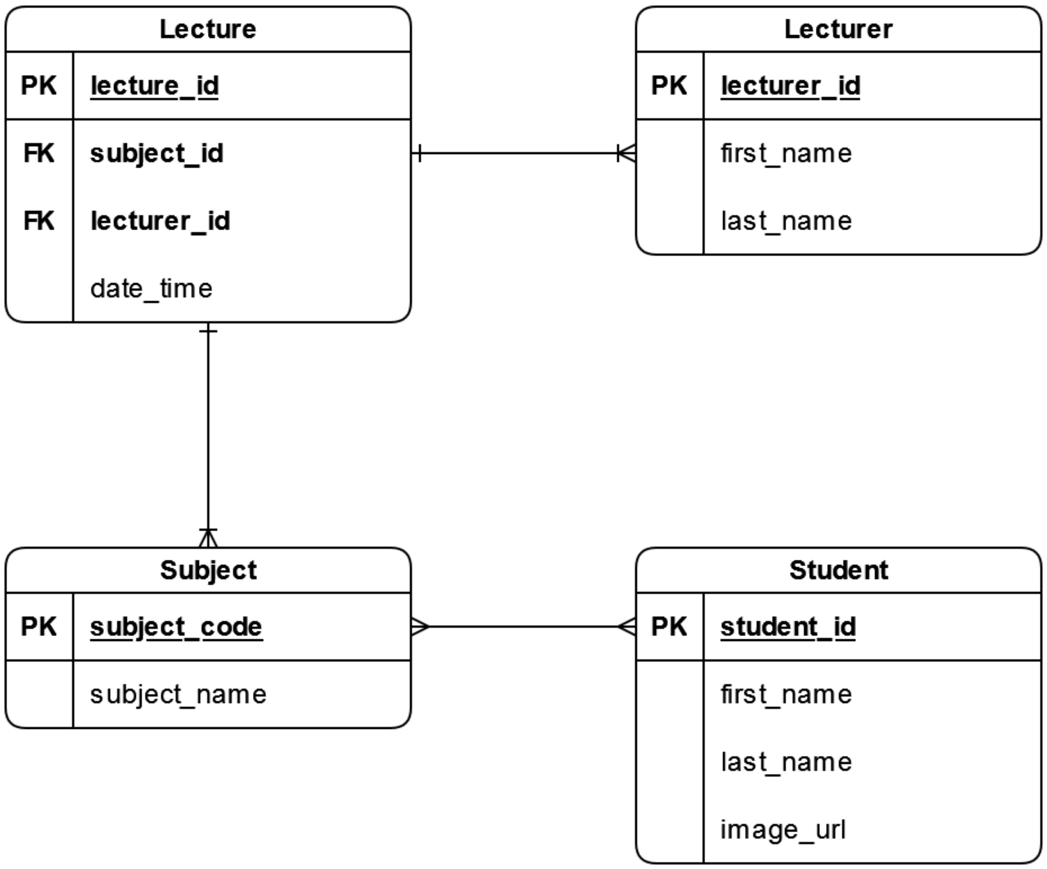
JPype Library:
Alternatively, another approach we've looked into using is the JPype library, which allows us to call Python functions directly from Java (MyExamCloud 2023). This method provides closer integration between Python and Java components, allowing for smoother communication and interaction. However, this requires additional setup and configurations, such as the installation and administration of the JPype library and Python environment. While increasing flexibility and control over the integration process, it may present difficulties that must be properly controlled.

In summary, both approaches provide feasible ways to integrate Python with Java Spring Boot. The decision to choose between them is determined by criteria such as the complexity of interactions and performance requirements. By carefully considering these factors, our team can determine the best method to achieving smooth integration and efficient collaboration between Python and Java components within the Spring Boot framework for Lectoro.

DATABASE DESIGN

The database serves as the backbone of the system, storing essential data related to student attendance, course information, and system configurations. Through meticulous planning and analysis, we aim to define an efficient and scalable database schema that meets the project's requirements. By outlining the structure, relationships, and constraints of the database, we lay the groundwork for seamless data storage, retrieval, and manipulation within the Lectoro system.

Diagram 3. Database Schema



Lecturer:
stores the first and last name of the lecturer that intends to use the attendance system.

Subject:
stores the name of the subject of the lecture.

Lecture:
stores the date and time of the arrival of students. The Lecture entity is linked with Lecturer and Subject by their IDs as foreign keys to maintain data integrity and prevent errors.

Student:
stores student information including their first name, last name and the URLs of the file paths for student photos. Multiple student photo URLs are stored in an array and are used as references for the facial recognition system to detect student's faces.

BACK-END DESIGN CONCLUSION

In conclusion, the Lectoro's backend design includes a thorough architecture that is optimised for efficient data handling and system functionality. The database design is critical for keeping important information on lecturers, subjects, lectures, and students while also assuring data integrity and accessibility. By structuring entities such as Lecturer, Subject, Lecture, and Student with appropriate attributes and relationships, we establish a solid foundation for the automatic attendance system.

Spring Boot's monolithic architecture allows for a more streamlined approach to development (*Spring Modulith* 2024), which simplifies setup and deployment operations. Furthermore, Spring Boot's flexibility enables possible scalability via future restructuring into microservices (*Spring | Microservices* 2024), assuring responsiveness to changing requirements. And the incorporation of Python characteristics into Java Spring Boot shows potential for improving system capabilities, particularly when integrating facial recognition features. Overall, the backend design represents a cohesive and well-considered framework that aligns with project objectives and sets the stage for the successful implementation for Lectoro.

DEVELOPMENT ENVIRONMENT

Building Lectoro as a robust facial recognition system requires a well-defined development environment that fosters efficient coding practices, ensuring application scalability, and simplifying future maintenance. This project meticulously selects a suite of technologies to achieve these goals. This section delves into the core components of our development environment, highlighting the rationale behind each selection and its contribution to the overall project success.

We'll specify the chosen programming languages, frameworks, libraries, IDE options, and DBMS. Each element plays a crucial role in efficiently building and deploying the facial recognition system, ensuring its functionality and scalability meet project requirements.

PROGRAMMING LANGUAGES

Python has been chosen as the primary development language due to its extensive ecosystem of libraries specifically designed for computer vision tasks (*7 Best Computer Vision Libraries in Python* 2024), including facial recognition. Furthermore, Python's well-known readability simplifies code maintenance and future modifications.

To meet project timelines, a pre-trained facial recognition model will be integrated, utilising Python libraries. Developing a custom model from scratch would be a resource-intensive endeavour, exceeding the current time constraints given.

The backend framework for the web application will leverage Java due to its established reputation for high performance and scalability. Additionally, Java's seamless compatibility with the Spring Boot framework allows for rapid application development and deployment.

FRAMEWORKS

Spring Boot is renowned for its ability to deliver robust and scalable applications. It optimises resource utilisation and streamlines application execution (*Spring Modulith* 2024), ensuring the web application can meet current and future performance demands. Spring Boot streamlines application development through its autoconfiguration capabilities. This eliminates the need for extensive manual configuration, allowing developers to focus on core functionalities. Additionally, Spring Boot applications are packaged as self-contained units, simplifying deployment processes. Spring Boot promotes code readability and maintainability with its focus on convention over configuration. This intuitive approach fosters faster development cycles and facilitates ongoing application maintenance.

PYTHON LIBRARIES

Core Libraries:
OpenCV library provides a foundation for computer vision tasks (*OpenCV* 2024), including image manipulation, feature detection, and object tracking. It's commonly used for pre-processing images before feeding them into a facial recognition model.

Moreover, NumPy library offers powerful array manipulation capabilities (*NumPy* 2023) essential for numerical computations used in facial recognition algorithms.

"face_recognition" library simplifies facial recognition tasks by building upon OpenCV and Dlib (*face-recognition* 2020). It provides functions for face detection, landmark localisation, and face image encoding for comparison.

Additional Libraries:
Scikit-learn library provides a comprehensive suite of machine learning tools (*scikit-learn: machine learning in Python* 2024). While not essential for basic facial recognition, it might be useful if we plan to explore custom model training or advanced image processing techniques.

Matplotlib/Seaborn libraries are helpful for data visualization (*An introduction to seaborn* 2024) if we want to analyse or present the performance of our facial recognition system.

These additional libraries are extensive in their capabilities to the development for our project, but will ultimately only be used depending on our needs during development.

INTERGRATED DEVELOPMENT ENVIRONMENT (IDE)

Choosing the appropriate Integrated Development Environment (IDE) is critical for efficient and productive software development. In this part, we discuss the IDEs used to construct Lectoro and the facial recognition attendance system. Our choice of IDEs is made after thorough analysis of issues such as performance, functionality, and project needs.

VSCode:
VS Code boasts a fast and resource-efficient design, making it ideal for various development scenarios (Microsoft 2021). Furthermore, its extensive extension marketplace allows for tailored functionality through plugins catering to specific languages and frameworks (Microsoft 2021).

Also, it offers excellent built-in support for Python development, including syntax highlighting, code completion, and debugging tools (Microsoft 2021). Additional extensions can further enhance these capabilities, providing features like linting and unit testing integration. With a large and active community that contributes extensions, tutorials, and troubleshooting resources, making it easy to find solutions and support.

IntelliJ:
With its powerful code editing features, IntelliJ IDEA offers refactoring tools, intelligent code completion, and on-the-fly code analysis (JetBrains 2024). These features significantly improve development speed and code quality.

For projects leveraging Spring Boot (like our application), IntelliJ IDEA offers dedicated tooling for seamless integration. This includes auto-configuration suggestions (*IntelliJ IDEA Help* 2024), Spring-specific debugging capabilities, and a streamlined development experience (*IntelliJ IDEA Help* 2024).

IntelliJ IDEA provides comprehensive project management tools alongside excellent code navigation features. This enhances developer productivity by facilitating navigation within large codebases and managing project dependencies effectively.

PyCharm:
Another option is PyCharm, it offers a powerful and feature-rich environment specifically designed to streamline Python development (JetBrains 2024). Its integration with scientific libraries, debugging tools, and project management capabilities (JetBrains 2024) make it a strong contender for this facial recognition project. Carefully evaluating project needs and developer preferences against PyCharm's offerings will help determine if it's the optimal IDE choice.

DATABASE MANAGEMENT SYSTEM (DBMS)

While relational databases like MySQL or PostgreSQL are established solutions, their rigid schema structure presents limitations for this project. Facial recognition data can be diverse, and evolving attendance record forms necessitate flexibility. MongoDB's schema-less document model (*MongoDB: The Developer Data Platform* 2024) overcomes this hurdle by allowing for the storage of various data types within a single document. This adaptability ensures seamless integration of new data points without schema modifications (*Schemaless Database* 2024), a key benefit when future requirements change.

Furthermore, traditional relational databases often require separate file systems or complex operations to handle binary data like facial recognition images. MongoDB's Binary Data (BinData) type streamlines this process by efficiently storing binary data alongside other information within the same document (Arti Technologies 2024). This simplifies data management and retrieval, eliminating the need for additional configurations or workarounds.

Finally, scalability is a crucial consideration. As the facial recognition system accumulates data, efficient storage and retrieval become paramount. MongoDB excels in this area, with horizontal scaling capabilities that allow for effortless data distribution across multiple servers (*MongoDB: The Developer Data Platform* 2024). This ensures optimal performance even with growing datasets, surpassing the limitations of many other Database Management Systems.

HOSTING AND DEVELOPMENT PLATFORM

The automatic attendance system will be deployed using Amazon Web Services to take advantage of the various services and servers that AWS provides:

Amazon EC2 provides resizable compute capacity in the cloud (*Cloud Computing Services - Amazon Web Services (AWS)* 2024) and is well-suited for hosting the backend of our facial recognition app. We can choose the instance type based on our computing requirements, and EC2 offers flexibility in terms of operating systems, programming languages, and frameworks.

Amazon S3 can be used to store static assets (*AWS Amazon EC2 2023 - Compute Innovation Talk* 2023) such as images, video files, or trained machine learning models used in our facial recognition system. It offers high durability, scalability, and availability for storing large amounts of data.

DEPLOYMENT ENVIRONMENT

A well-planned deployment environment guarantees the smooth operation for Lectoro’s facial recognition system in a real-world setting. This section specifies the chosen deployment platforms and outlines the strategies employed to optimise performance, and scalability. We will delve into the specific services utilised, configuration considerations, and best practices for maintaining a robust deployment environment.

TESTING ENVIRONMENT

The functionalities, performance and databases of Lectoro’s automatic attendance system will be initially tested locally to ensure no bugs or errors are present before being deployed in AWS. Within the testing environment, the facial recognition system will use laptop webcams and possibly other external cameras to ensure reliability between various hardware. Photos and student data just from our team members will be used to test facial recognition, other functionalities and also to not breach any privacy regulations during the phases of our project. Fake sample students may also be created to test performance and reliability of the system, especially during high load.

Hardware:

Personal computers and / or provided computers used for the development of Lectoro.

External webcam or integrated laptop webcam will be used to develop and test the facial recognition system.

Software:

Integrated Development Environment for testing and debugging errors.

Version control will be enforced using Git and GitHub to allow our developers to collaborate and keep track of changes in code.

STAGING ENVIRONMENT

The Staging Environment is an important stage between development and production, allowing for thorough testing and validation of Lectoro and its intended functionalities prior to deployment. In our deployment plan, the staging environment will be hosted on Amazon Web Services (AWS), taking use of its strong infrastructure and scalable services (*Cloud Computing Services - Amazon Web Services (AWS)* 2024) to imitate real-world usage situations and verify the system's dependability and performance.

Prior to deployment, the automatic attendance system will be thoroughly tested and will be designed to assess the system's functionality, performance, and dependability under a variety of scenarios. To effectively imitate real-life application usage, the testing procedure will take place in a spare lecture room that is similar to the actual context in which the system will be deployed. This assures that the system works as intended and is capable of handling the demands of live usage, such as real-time attendance data processing and smooth user interaction.

During testing, all aspects of Lectoro's functioning will be extensively evaluated, including facial recognition accuracy, attendance tracking dependability, and user interface responsiveness. Any faults or anomalies discovered during testing will be resolved immediately to guarantee that the system satisfies the relevant quality and performance criteria.

In addition to functional testing, the staging environment allows for performance optimisation. Performance parameters like as response times, resource utilisation, and scalability will be monitored and analysed to identify possible bottlenecks and opportunities for improvement. This proactive strategy guarantees that the system can successfully handle the projected workload while also scaling to meet future demand.

By leveraging the staging environment for thorough testing and validation, we can confidently deploy Lectoro with the assurance that it meets the intended capabilities with the highest standards of quality, reliability, and performance.

PRODUCTION ENVIRONMENT

The Production Environment is the result of meticulous planning and preparation, with the transition of Lectoro moving from development to actual deployment. In this section, we look at the hardware and software components that comprise the foundation of the production environment, ensuring that the system runs smoothly and performs optimally.

Hardware:
A dedicated server instance on AWS EC2 will be chosen based on the expected number of users and processing needs. This server will run the backend application using Java and communicate with the MongoDB database.

High-definition cameras will be installed at strategic locations in the lecture room to capture student faces clearly. The number (ideally just 1) and positioning of cameras will depend on the room size and layout to minimise occlusion (objects blocking faces).

Depending on the desired workflow and future projects, the QR code system, touch-screen kiosks or dedicated computers might be installed for students to identify themselves through the system (entering a PIN or student ID).

Software:
The backend application Java connected with Spring Boot will be deployed on the EC2 server. The facial recognition model (Python libraries) will be integrated with the backend for real-time recognition.

Additional Considerations:
Proper lighting conditions in the lecture room are very important for accurate facial recognition. Consider adjustable lighting or ensure consistent ambient light to avoid recognition issues.

A strong internet connection is essential for communication between the cameras, server, and database.

As we get deeper into the production environment, we will examine the complexities of hardware selection, software integration, and environmental issues, with the ultimate objective of developing a strong and dependable foundation for Lectoro. Through the demands of preparation and attention to detail, we want to provide a seamless transition from development to live deployment, providing lecturers and administrators with a strong tool for effective attendance management at UOW.

FRONT-END DESIGN

In crafting Lectoro as an Automatic Attendance System, the front-end serves as the gateway to a seamless user experience. We've opted for a dynamic HTML webpage, enriched with CSS and JavaScript functionalities, complemented by a framework tailored for rapid development. While exploring integration with UOW's Moodle platform, resource constraints led us to focus on an independent webpage solution for now. This report explores our front-end architecture, communication with the back-end, stakeholder scenarios, and our rationale behind chosen technologies and design principles. Our aim is to prioritise scalability, maintainability, and user-centric design to enhance the overall user experience.

COMMUNICATION WITH BACK-END

The front-end interacts with the back-end through RESTful APIs, enabling seamless data exchange (*What is RESTful API? - AWS* 2024) for various functionalities. When a user interacts with the front-end, such as logging in, accessing course information, or managing attendance records, the front-end sends requests to the back-end API endpoints. These requests contain relevant data or parameters required for the operation. Upon receiving the request, the back-end processes the data, performs necessary actions, and sends back a response, which the front-end interprets and displays to the user. This communication flow ensures efficient data transfer and enables real-time updates and interactions within the application.

Administrators:
Manage Subjects: Administrators can add, edit, delete, and assign subjects to lecturers using dedicated administrative interfaces. They have full control over the subjects offered within the system.

Student Enrolment: Administrators can manage student enrolment for each subject, including adding or removing students from course rosters.

Attendance Data Access: Administrators have access to comprehensive attendance data and reports for each subject, allowing them to monitor attendance trends, generate reports, and analyse attendance patterns.

Lecturers:
View Assigned Subjects: Lecturers can view the subjects assigned to them, along with the list of enrolled students for each subject.

Attendance Management: Lecturers can access attendance data specific to their assigned subjects, allowing them to track student attendance, view attendance records, and monitor student participation.

Report Generation: Lecturers can generate attendance reports for their subjects, facilitating analysis and insights into attendance trends and patterns.

ARCHITECTURE AND TECHNOLOGIES

The front-end architecture adopts a modular and component-based approach to facilitate scalability, maintainability, and code reusability. It leverages HTML, CSS, and JavaScript as foundational technologies for building the user interface. While React.js was initially considered, the decision is now made to forego its use to accommodate team members with varying levels of proficiency. Instead, we've opted for Tailwind CSS, a utility-first CSS framework.

Tailwind CSS provides a robust set of utility classes that enable precise control over styling (*Tailwind CSS* 2020) without the need for custom CSS. To further extend Tailwind's capabilities with pre-designed components, we've integrated Flowbite, a plugin that enhances Tailwind projects with a library of ready-to-use UI components. This combination allows for rapid development and consistent styling across the website (Themesberg 2019), empowering our team to efficiently create and maintain a cohesive user interface while balancing simplicity and flexibility.

Tailwind CSS Utility Classes:
Tailwind CSS offers a comprehensive array of utility classes for precise control over styling. For instance, applying the bg-blue-500 class sets the background colour to a shade of blue, while px-4 py-2 adds padding to all sides of an element.

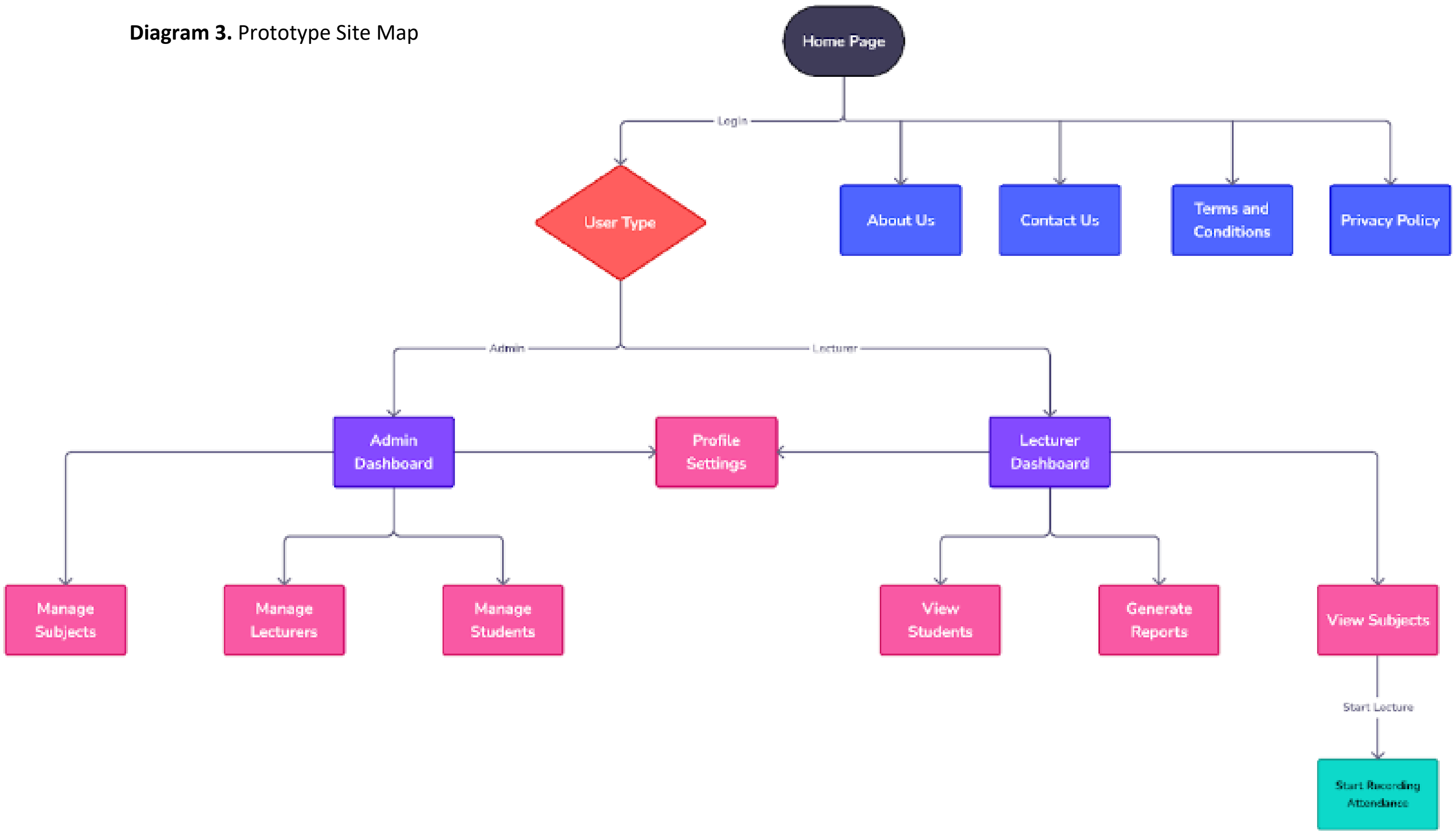
Flowbite Integration for Pre-designed Components:
By integrating Flowbite, developers gain access to a repository of pre-designed UI components that seamlessly augment Tailwind projects. With Flowbite, creating interactive elements like dropdown menus becomes as simple as adding attributes such as data-dropdown-toggle to trigger dropdown functionality.

Rapid Development and Consistent Styling:
The fusion of Tailwind CSS and Flowbite streamlines development workflows, enabling swift creation of a cohesive user interface. This combination facilitates rapid iteration and ensures consistency in styling across the website, empowering the team to deliver a polished user experience efficiently.

PROTOTYPE DIAGRAMS

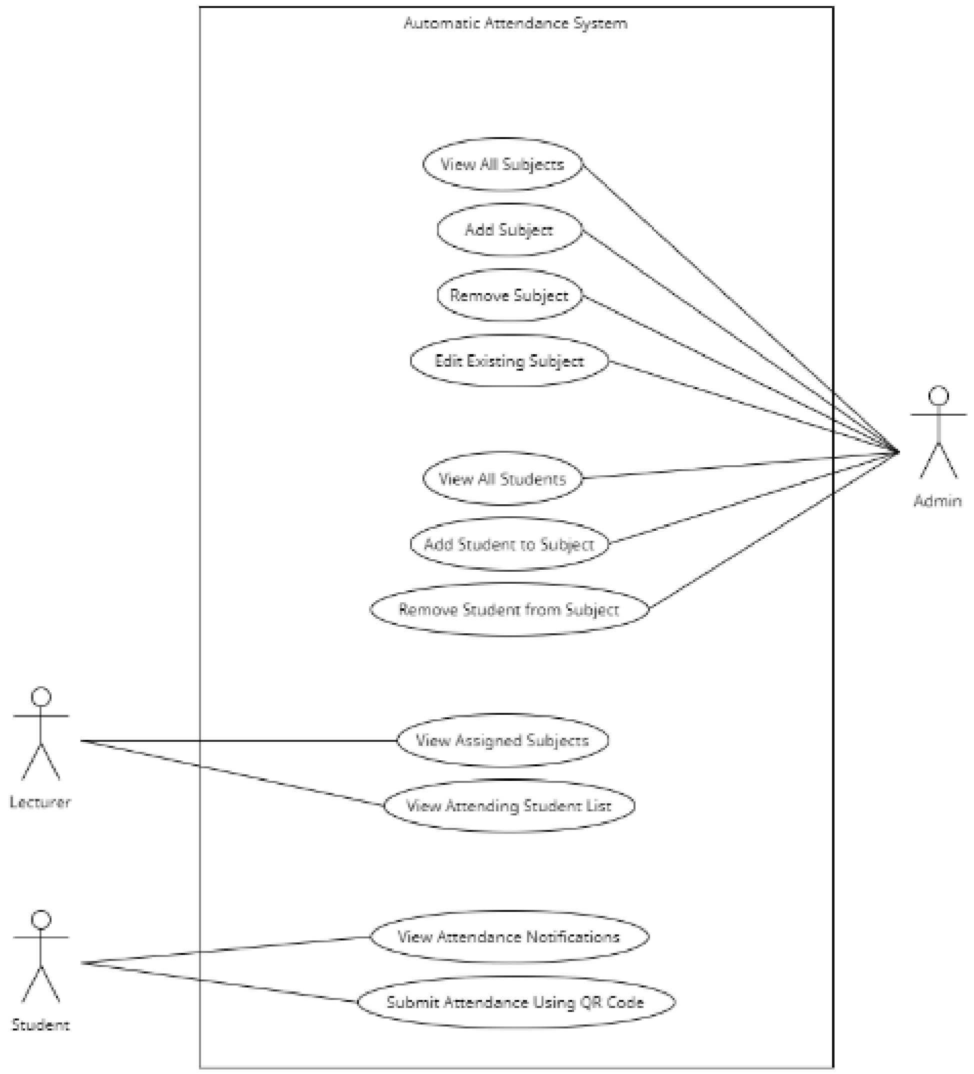
For the visualisation of our front-end design, we've created prototype diagrams, which are critical tools for conceptualising, developing, and explaining the structure and functioning of the Lectoro system. We look at three main diagrams: site maps, use case diagrams, and entity relationship diagrams. These diagrams give a thorough overview of the system's architecture, user interactions, and data relationships, providing significant insights into its design and functionality. The purpose of examining these prototype diagrams is to illustrate the logical flow, user journeys, and data management procedures inherent in the Lectoro system's front-end design.

Diagram 3. Prototype Site Map



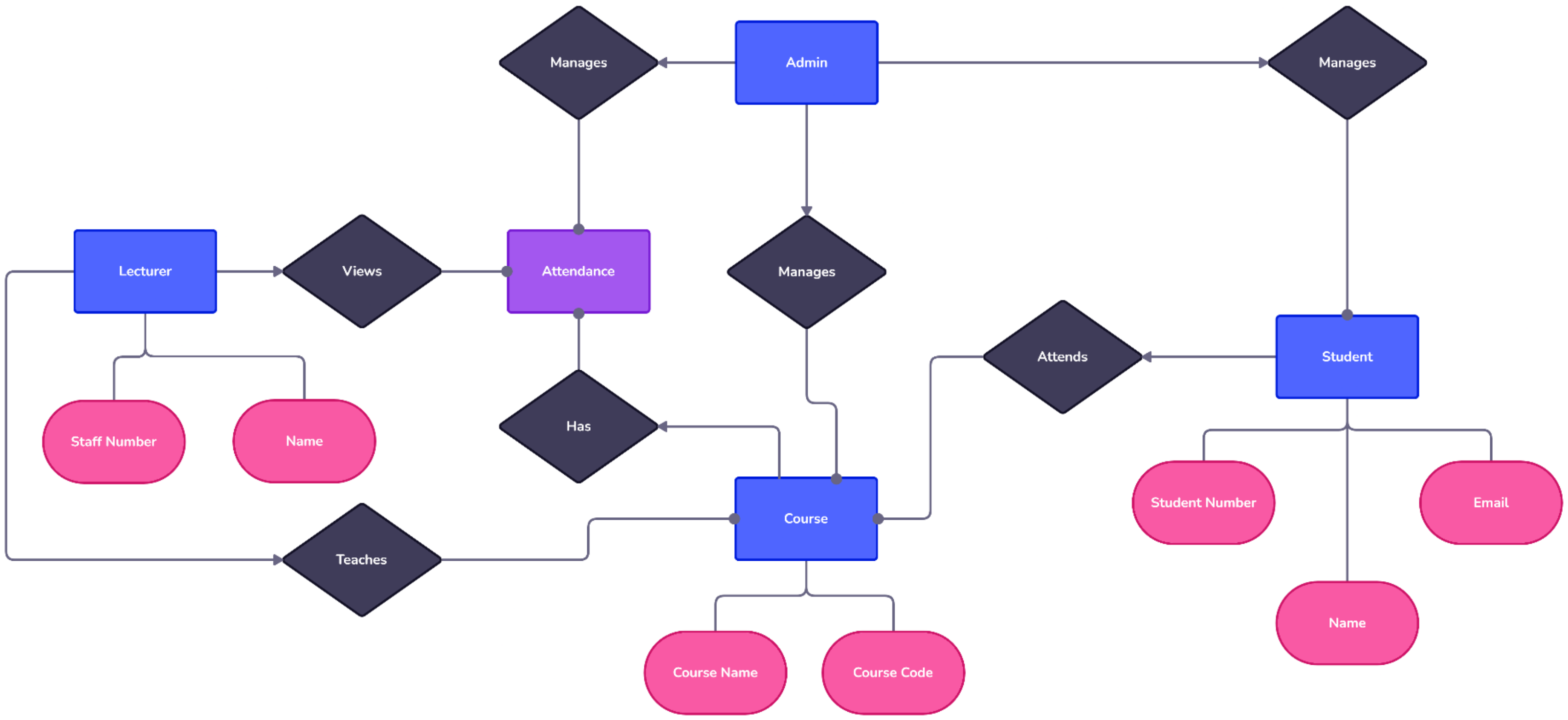
The prototype Site Map visually represent the user journeys, navigation flow, and interaction pathways within the application. It outlines the sequence of actions users take to accomplish specific tasks, such as logging in, accessing course information, managing attendance records, and generating reports. By providing a visual representation of the application's functionality and user experience, the Site Map serves as a valuable reference for development, guiding the implementation of features and interactions to align with user expectations and requirements for Lectoro.

Diagram 4. Use Case Diagram



This is a use case diagram for our system. Administrators have privileges to view and edit the lists of subjects and students. Lecturers can view their assigned students for each of their subjects, as well as viewing their attendance. Students can receive notifications for their attendance, as well as submit said attendance using a QR Code.

Diagram 5. Entity Relationship Diagram



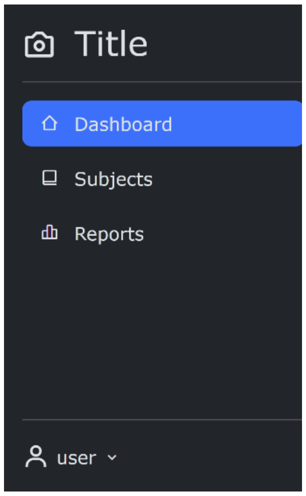
This is an entity relationship diagram of the system. Students attend their courses (which are logged), which are taught by lecturers. Lecturers can view the attendance of their students. Administrators are able to view and edit the entire system, including courses, attendance and students

USER INTERFACE DESIGN

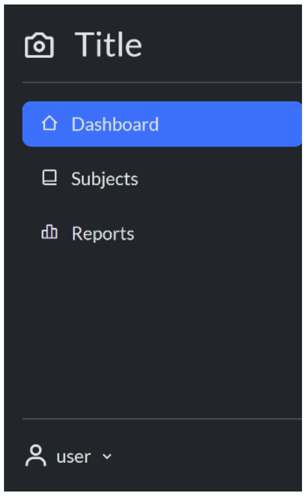
The user interface design prioritises simplicity, intuitiveness, and accessibility to enhance user experience. Clear and intuitive navigation structures guide users through the application, ensuring easy access to essential features and functionalities. Minimalistic design elements, such as clean layouts, legible typography, and intuitive iconography, contribute to a clutter-free interface that promotes user engagement and task completion. Additionally, the user interface design adheres to accessibility standards, ensuring compatibility with assistive technologies and enabling inclusivity for all users, drawing inspiration from intuitive design principles, including influences from platforms like Jibble.

DESIGN BRAINSTORM

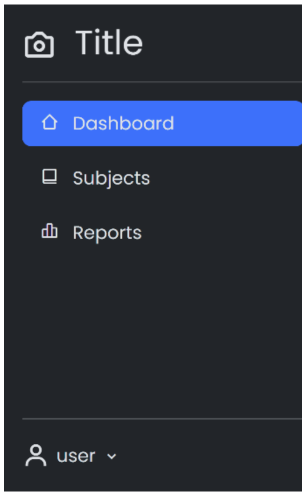
Verdana



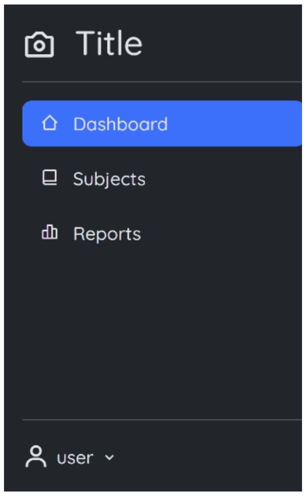
Lato



Poppins



Quicksand



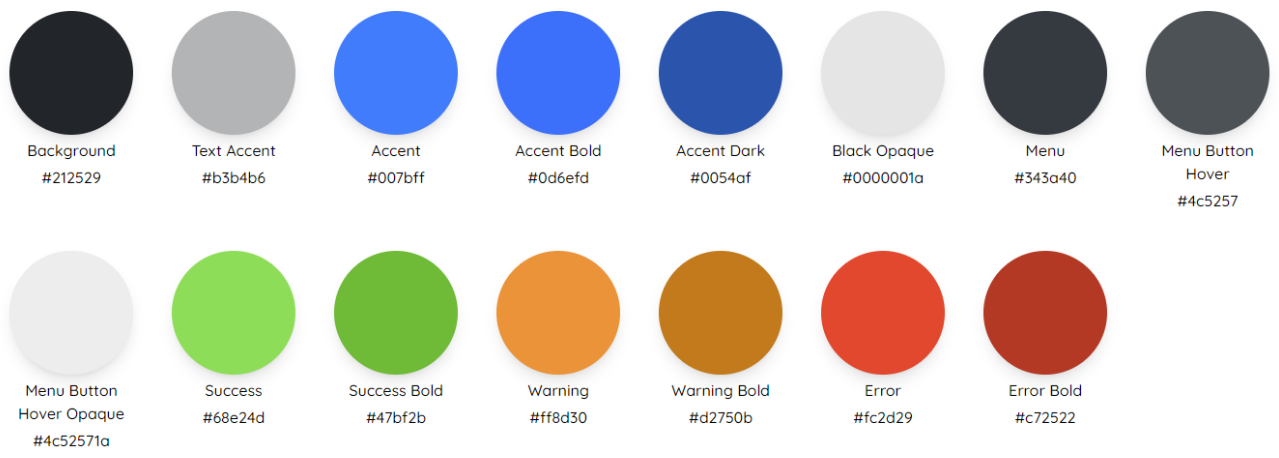
The fonts Verdana, Lato, Poppins, and Quicksand are evaluated for readability and visual cohesion. Verdana is used as the control font, establishing a uniform benchmark. Lato and Quicksand both demonstrate sharp lettering that improves legibility, but Quicksand has a minor advantage due to its softly rounded letters, which result in a smoother reading experience. While Poppins shares this clarity, its lettering appears disorganised, diminishing its overall appeal. Quicksand, on the other hand, emerges a refined aesthetic that virtually no other font can equal, making it the favoured choice due to its harmonious balance of clarity and sophistication.

PRODUCT BRAINSTORM

After considering various options and exploring names that encapsulate the essence of our automatic attendance system, we have chosen 'Lectoro' as our product name. Derived from 'lecturer' and 'technology,' Lectoro perfectly represents our platform's focus on facilitating lecture management and leveraging innovative technological solutions. The name resonates with our goal of providing a streamlined, efficient, and intuitive experience for both administrators and lecturers. Lectoro embodies our commitment to revolutionizing attendance tracking in educational settings while maintaining a memorable and distinctive identity.

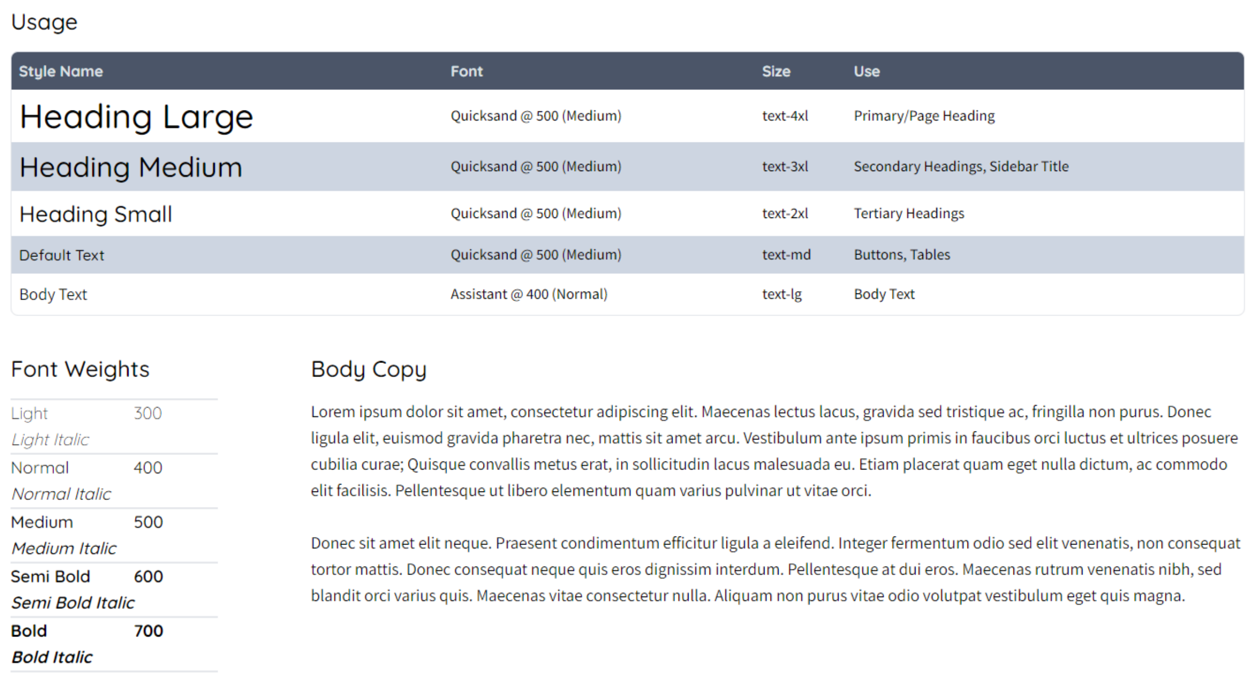
STYLE GUIDE

Figure 3. Style Guide Colour Palette



Background	#212529
Text Accent	#b3b4b6
Accent	#007bff
Accent Bold	#0d6efd
Accent Dark	#0054af
Black Opaque	#0000001a
Menu	#343a40
Menu Button Hover	#4c5257
Menu Button Hover Opaque	#4c52571a
Success	#68e24d
Success Bold	#47bf2b
Warning	#ff8d30
Warning Bold	#d2750b
Error	#fc2d29
Error Bold	#c72522

Figure 4. Style Guide Typography



Usage

Style Name	Font	Size	Use
Heading Large	Quicksand @ 500 (Medium)	text-4xl	Primary/Page Heading
Heading Medium	Quicksand @ 500 (Medium)	text-3xl	Secondary Headings, Sidebar Title
Heading Small	Quicksand @ 500 (Medium)	text-2xl	Tertiary Headings
Default Text	Quicksand @ 500 (Medium)	text-md	Buttons, Tables
Body Text	Assistant @ 400 (Normal)	text-lg	Body Text

Font Weights

Light	300
Light Italic	
Normal	400
Normal Italic	
Medium	500
Medium Italic	
Semi Bold	**600**
Semi Bold Italic	
Bold	**700**
Bold Italic	

Body Copy

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lectus lacus, gravida sed tristique ac, fringilla non purus. Donec ligula elit, euismod gravida pharetra nec, mattis sit amet arcu. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Quisque convallis metus erat, in sollicitudin lacus malesuada eu. Etiam placerat quam eget nulla dictum, ac commodo elit facilisis. Pellentesque ut libero elementum quam varius pulvinar ut vitae orci.

Donec sit amet elit neque. Praesent condimentum efficitur ligula a eleifend. Integer fermentum odio sed elit venenatis, non consequat tortor mattis. Donec consequat neque quis eros dignissim interdum. Pellentesque at dui eros. Maecenas rutrum venenatis nibh, sed blandit orci varius quis. Maecenas vitae consectetur nulla. Aliquam non purus vitae odio volutpat vestibulum eget quis magna.

Figure 5. Style Guide Inputs

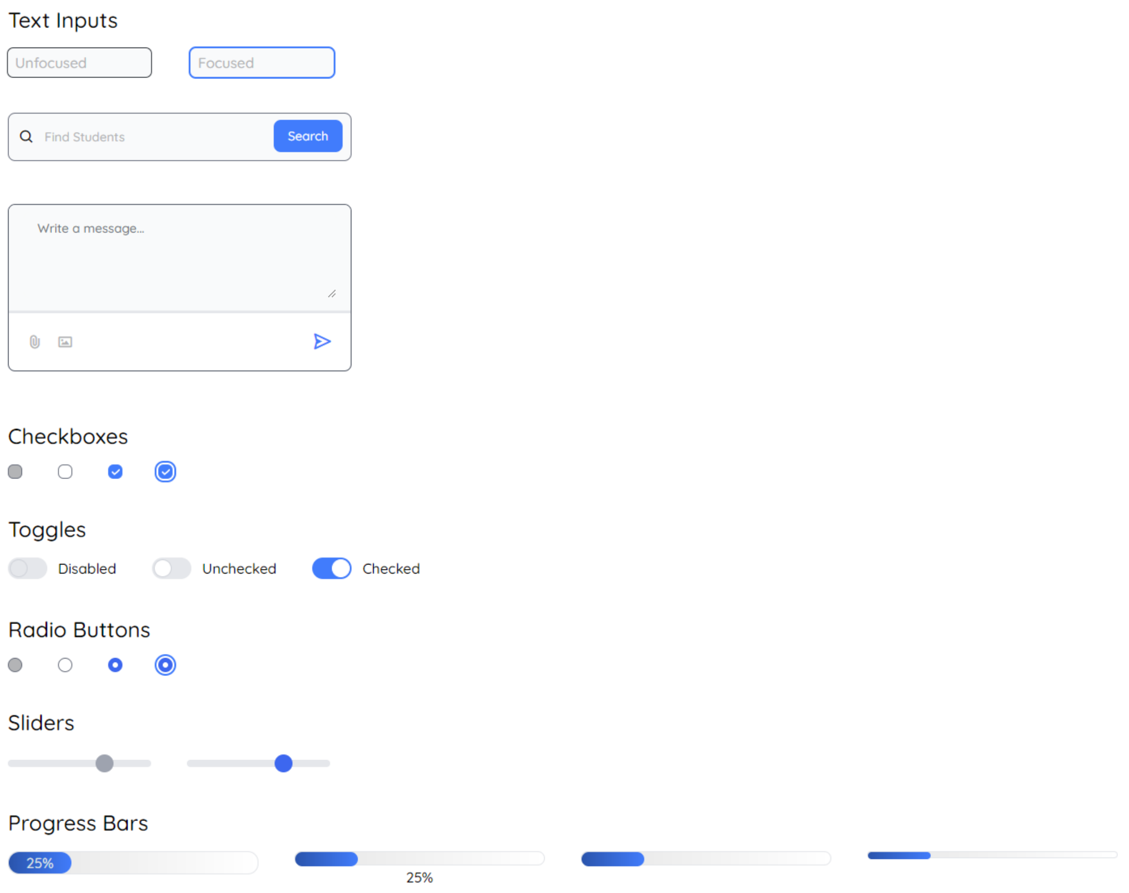


Figure 6. Style Guide Menus

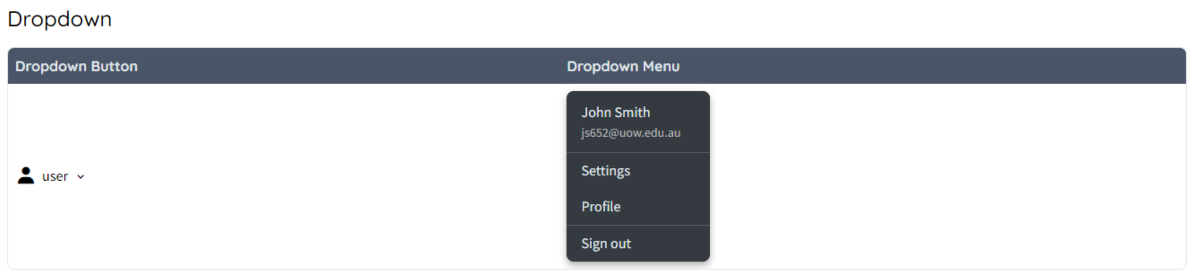


Figure 7. Style Guide Buttons

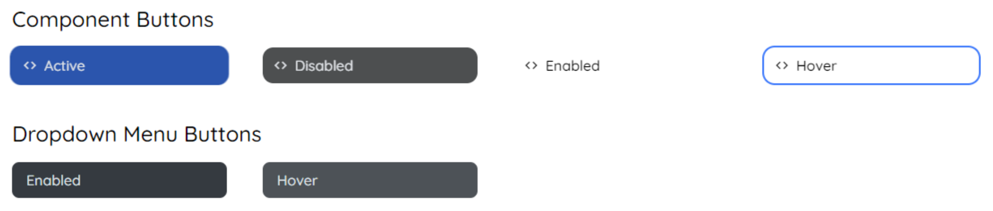


Figure 8. Style Guide Alerts

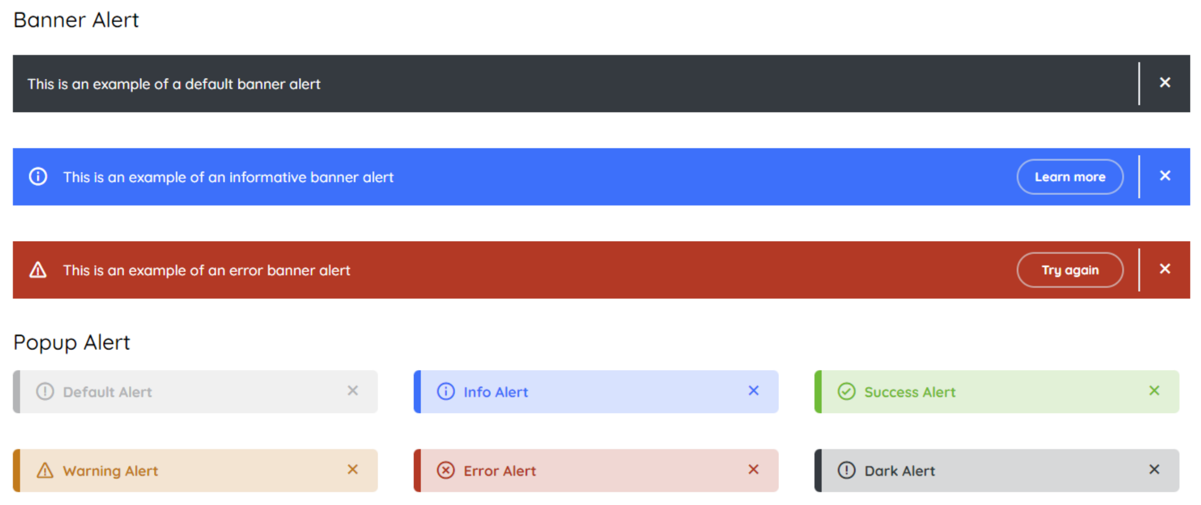
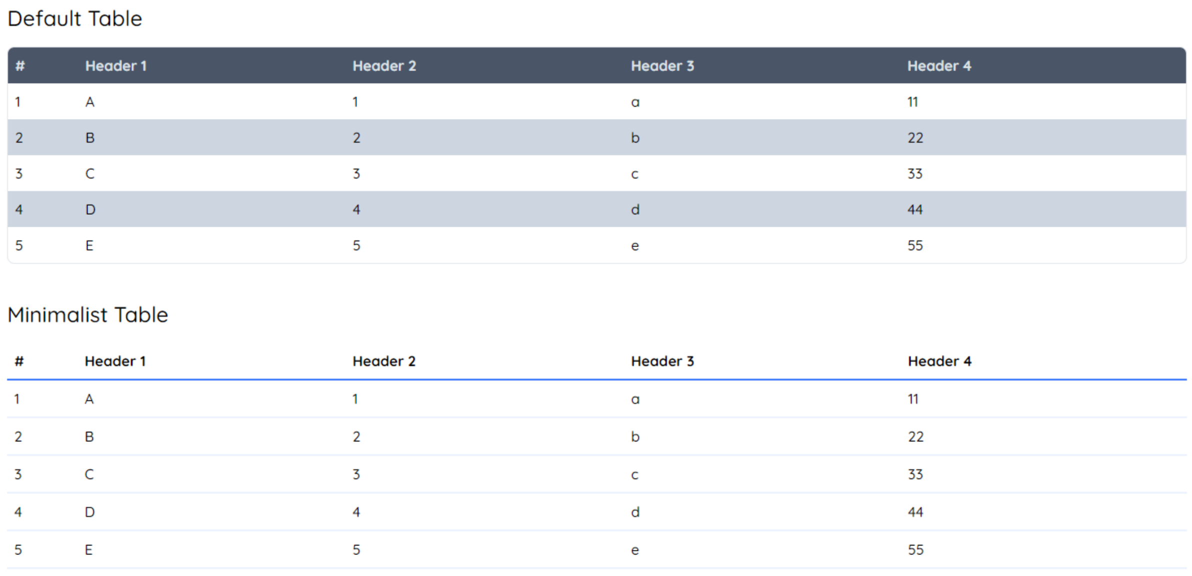

Banner Alert
This is an example of a default banner alert
This is an example of an informative banner alert
Learn more
This is an example of an error banner alert
Try again
Popup Alert
Default Alert
Info Alert
Success Alert
Warning Alert
Error Alert
Dark Alert

Figure 9. Style Guide Tables



Default Table

#	Header 1	Header 2	Header 3	Header 4
1	A	1	a	11
2	B	2	b	22
3	C	3	c	33
4	D	4	d	44
5	E	5	e	55

Minimalist Table

#	Header 1	Header 2	Header 3	Header 4
1	A	1	a	11
2	B	2	b	22
3	C	3	c	33
4	D	4	d	44
5	E	5	e	55

The style guide serves as a foundational document, detailing specific guidelines and specifications for the visual design of the Lectoro system. We improve usability and brand recognition by carefully selecting typefaces, colours, and design components that are consistent and coherent across all user interfaces. The style guide's rigors approach to design decisions ensures a consistent and professional user experience, displaying our dedication to quality and attention to detail. As we adhere to these standards throughout the development process, we enable our team to create visually beautiful and user-friendly interfaces that are consistent with the project's objectives and meet the expectations of our stakeholders.

FRONT-END CONCLUSION

To conclude, the front-end design of Lectoro, our Automatic Attendance System, reflects our commitment to providing a seamless and user-centric experience. We prioritise scalability, maintainability, and user happiness by using dynamic HTML webpages that include CSS and JavaScript functionality.

Our front-end architecture enables effective communication with the back-end via RESTful APIs, resulting in seamless data interchange (*What is RESTful API? - AWS* 2024) for diverse functionalities. Administrators benefit from extensive management tools such as subject administration, student enrolment management, and access to attendance data and reports. Similarly, instructors benefit from functions aimed to their needs, such as viewing assigned subjects, tracking attendance, and creating reports. Our front-end architecture is modular and component-based, with basic technologies such as HTML, CSS, and JavaScript. While React.js was first explored, we chose Tailwind CSS and Flowbite to handle our team's different skill levels, allowing for speedy development and uniform styling (Themesberg 2019) across the site.

Prototype diagrams, which include site maps, use case diagrams, and entity relationship diagrams, provide useful information about the system's design, user interactions, and data linkages. These diagrams are critical in conceptualising, creating, and describing the structure and operation of the Lectoro system's front-end design. Our user interface design focuses on simplicity, intuitiveness, and accessibility, with obvious navigation structures, minimalistic design elements, and respect to accessibility guidelines. The style guide is a foundational document that outlines particular principles and specifications for visual design, assuring uniformity and professionalism across all interfaces.

To summarise, our front-end design for Lectoro exemplifies our commitment to providing a user-friendly and visually appealing solution that satisfies our stakeholders' expectations and improves the overall user experience.

STAKEHOLDER AND SYSTEM SCENARIOS

To guarantee the effective deployment and operation of the Lectoro Automatic Attendance System, it is critical to understand the roles, responsibilities, problems, and strategies of key stakeholders. By detailing the interdependencies, prospective actions, and impact assessments, we hope to foresee alternative outcomes. Furthermore, we present a full system scenario that includes preconditions, basic and alternate flows, and postconditions to demonstrate how Lectoro works in real-world classroom situations. Through this thorough approach, we hope to guarantee that Lectoro satisfies stakeholder expectations, handles privacy issues, and allows for effective attendance monitoring throughout educational settings in UOW.

Administrators:
Responsibilities: As a major stakeholder and the secondary user, they control the management and function of Lectoro for attendance monitoring, the administration is responsible for ensuring that the system accomplishes operational goals while also harmonising with the university's wider strategic objectives. This involves improving academic performance tracking, optimising administrative operations, and ensuring adherence to educational standards.

Challenges: Despite their authority, the administration must deal with ethical, legal, and societal issues. There is a fundamental need to balance Lectoro's capabilities with privacy concerns, particularly those raised by students. Engaging these groups is critical for gathering input and instilling a feeling of inclusiveness and openness.

Strategy: To successfully manage Lectoro, our team at NexTech Innovations intends to develop a comprehensive communication and engagement strategy that answers all stakeholders' concerns and expectations. This entails openly revealing how Lectoro operates, the data it gathers, how the data is protected, and how privacy is maintained. The objective is to gain confidence and support from UOW by demonstrating a commitment to ethical standards and privacy protection.

Lecturers:
Responsibilities: Lecturers are the main user and are responsible for learning and implementing Lectoro into their teaching practices. Their direct involvement with the system and firsthand knowledge of its operation put them in a strong position to give input on its efficacy and usability.

Challenges: Lecturers are first to ensure that Lectoro improves the educational experience without becoming a hindrance to efficient learning, which would ultimately result in returning to traditional attendance gathering techniques. They must also be prepared to handle any technological issues or errors that emerge and are flagged by the system, which will need proper training and assistance from our team or the IT department.

Strategy: To maximise the usage of Lectoro, lecturers can collaborate closely with our team, administrators or the IT department to tailor the system's capabilities to their individual teaching needs and lecture room dynamics. They may assist improvements for the system by actively offering thorough feedback. This will benefit both instructors and students.

INTERDEPENDENCIES

The success of the Lectoro Automatic Attendance System is dependent on a network of interdependencies among our important stakeholders, each with their own duties and responsibilities. The administration relies on our team at NexTech Innovations to oversee the proper operation of the facial recognition software and to include any extra features that may be necessary in the future. This partnership is critical to aligning the system with the university's changing demands and strategic goals. Furthermore, Lecturers rely on the administrators to ensure that the technologies run with no issues during lectures. This includes addressing any technological difficulties that may emerge and offering assistance to guarantee a successful teaching experience. Additionally, students rely on the administration to protect their personal information. It is critical that strong data protection procedures are in place to address privacy concerns and meet with legal obligations, creating trust and confidence among students. These interdependencies highlight the critical need of excellent communication, cooperation, and coordination among all stakeholders in ensuring the Lectoro system's successful installation and use.

POTENTIAL ACTIONS

Lecturers have an important role in integrating Lectoro into their teaching methods, and they may take a variety of actions to ensure that the system meets their educational objectives while respecting privacy concerns. One such option is for lecturers to seek changes or updates to the technology depending on their instructional needs. This might include tailoring some features like report crafting and internal messages to be updated to make sure the dynamics adhere to the lecturers needs, lecturers and addressing certain teaching requirements. Additionally, lecturers may express privacy concerns about the usage of facial recognition technology in the lecture room. To comfort themselves and their students, they may urge for more privacy safeguards or greater openness in data management procedures.

Students, on the other hand, have a say in how Lectoro is adopted and implemented in the academic setting. Students may opt to oppose the usage of facial recognition technology, especially if they are concerned about privacy or data security. Such protests might take the form of organised campaigns, petitions, or demonstrations aiming at raising awareness about the perceived hazards or ethical implications of face recognition technology in educational settings. Furthermore, students may demand non-facial recognition attendance tracking techniques. Their support for alternative solutions emphasises the necessity of examining several views and ensuring that technological implementations respect the rights and preferences of all stakeholders.

IMPACT ASSESSMENT

The benefits of implementing Lectoro include simplifying administrative processes and minimising the effort associated with manual attendance monitoring. Administrative personnel may save time and costs by automating the attendance process with facial recognition technology, rather than manually entering and reconciling data. This efficiency enhancement can boost productivity and allow lecturers, administrators and even students to focus on more strategic activities, thus improving the overall efficacy of administrative operations.

However, there are possible negative consequences that must be properly evaluated and mitigated. One major concern is the possibility of privacy breaches and data security risks linked with the gathering and storage of face recognition data. If sensitive student information is not effectively protected, it may be subject to unauthorised access or exploitation, thereby harming the university's image and resulting in legal implications. Furthermore, certain members of the university community, such as students and staff, may be opposed to the use of facial recognition technology. Concerns regarding monitoring, permission, and the ethical implications of face recognition in educational contexts may result in resistance or opposition to the use of Lectoro.

SCENARIOS AND CONTINGENCIES

In the best-case scenario, Lectoro runs smoothly as intended, with no objections from stakeholders, resulting in increased administrative efficiency and very precise attendance data. The system works smoothly with current processes in lecturers and management systems, increasing operational efficiency and offering useful insights into attendance trends and patterns. This scenario promotes widespread acceptance and implementation of Lectoro within UOW, establishing it as a crucial tool for improving academic management and student involvement in large scale lectures.

In contrast, the worst-case scenario involves strong opposition from students, and lecturers, resulting in a reconsideration of the decision to use Lectoro or potentially lead to legal action. Concerns regarding privacy, data security, and the ethical implications of face recognition technology may result in significant opposition to its implementation. This situation presents substantial hurdles to the proper deployment of Lectoro, necessitating a careful balance of legal, ethical, and social considerations in order to satisfy stakeholder concerns and avoid possible hazards.

It is critical in contingency planning to anticipate possible challenges. One method is to return to a normal traditional manual attendance monitoring system as a backup alternative for dealing with technological challenges or stakeholder resistance. Furthermore, having a contingency option allows students to control their level of interaction, respecting privacy choices and giving an alternative for those who are uncomfortable with face recognition technology. These solutions show proactive risk management and a willingness to address stakeholder concerns while preserving operational performance.

SYSTEM SCENARIO

The system scenario depicts Lectoro's operational workflow inside the UOW university context, including preconditions, basic flow, alternate flows, and postconditions. It includes interaction with the Lectoro's management system, camera setup, and gaining authorisation. The flow includes system initialisation, face detection, identity verification, attendance recording, error management, and shutdown, guaranteeing a thorough comprehension. It also covers late arrivals and system breakdowns, highlighting attendance accuracy and data security.

PRECONDITIONS

Integration:
Lectoro seamlessly integrates with the UOW's lecture room, ensuring efficient operational synergy.

Installation:
Cameras, equipped with facial recognition capabilities, are strategically installed across all lecture rooms for accurate facial recognition identification.

Communication:
Students and lecturers are thoroughly briefed about Lectoro, providing informed consent for the use of facial recognition technology, following comprehensive discussions on privacy and data handling protocols.

BASIC FLOW

System Initialisation:
Activation: Prior to the start of a lecture, Lectoro is activated, in which is confirmed by Lectoro's management systems confirming correct subject time, enrolled students, etc.

Camera Operation: Cameras begin capturing video footage of the lecture room entrance, preparing to identify entering students.

Face Detection:
Detection Process: As students enter, Lectoro's cameras detect and capture facial images.

Image Processing: These images are immediately processed, and facial features are extracted and compared against the student face database maintained by the administrators.

Identity Verification:
Matching Algorithm: Lectoro matches the processed facial data with pre-registered student IDs in Lectoro's database.

Attendance Marking: For each successful match, Lectoro records the student's entry time and marks them as present in the system.

Attendance Recording:
Data Update: Attendance records are automatically updated and synchronised within Lectoro's management system.

Data Access: Lecturers and Administrators can view real-time attendance data directly on their connected devices, such as tablets or laptops.

Error Handling:
Unrecognized Faces: If Lectoro fails to recognise a face or matches it incorrectly, the system flags the event for manual review.

Manual Verification: Lecturers are notified of which students aren't accounted for by the facial recognition system. Lectoro during the lecture will send out notifications to these specific students, who if are attending the lecture can scan a QR code to manually account for their own attendance. Otherwise the lecturer can manually verify attendance if necessary.

System Shutdown:
Session Closure: At the conclusion of the lecture, the Lecturer officially closes the attendance session in Lectoro, and the system finalises and secures the attendance data for that session.

ALTERNATIVE FLOWS

Late Arrivals:
Continuous Scanning: Lectoro remains active throughout the lecture, continuously scanning for new faces. Late-arriving students are detected and recorded as soon as they enter the lecture room with a note stating their late arrival time.

Leaving Early:
Continuous Scanning: Since Lectoro remains active throughout the lecture, if a student leaves early during a lecture, Lectoro will detect the student as soon as they leave the lecture room with a note stating their early leave time.

System Failure:
Manual Override: In the event of a system error, such as camera malfunction or software issues, lecturers can revert to manual attendance tracking, or students can scan the QR codes.

Error Notification: The system automatically alerts the IT support team to address and resolve any technical issues.

POSTCONDITIONS

Attendance Accuracy: All students that have been correct accounted for by the facial recognition system are accurately marked and recorded into Lectoro.

Data Security: Attendance records are securely stored and maintained within Lectoro’s management system, accessible only to authorised personnel for academic and administrative purposes.

Finally, the system scenarios describe the complicated processes and protocols that govern Lectoro's operation within UOW's university lecture setting. This thorough structure, which defines preconditions, fundamental flows, alternate pathways, and postconditions, provides a clear understanding of how Lectoro works in real-world classrooms. Lectoro maintains data security and privacy requirements by ensuring accurate attendance monitoring through smooth integration, thorough installation, informed consent, and robust error handling procedures. These scenarios give a path for successful deployment by addressing probable problems and eventualities to create a dependable and efficient attendance management system.

CONCLUSION

To conclude, the Lectoro Automatic Attendance System design report is an extensive endeavour to conceptualise, develop, and execute a cutting-edge system with the goal of transforming attendance management in educational environments in lectures for UOW. Scalability, usability, and security are given top priority in the robust system architecture that we have created a comprehensive requirements analysis, an iterative design methodology, and careful consideration of stakeholder demands.

The joint efforts of our team at NexTech Innovations and our sponsor, Dr. Fenghui Ren, have resulted in a solution that, under our product Lectoro, not only satisfies the operational objectives for the university administration at UOW, but also improves the quality of teaching and learning for both lecturers and students. Through the use of state-of-the-art technology, industry best practices, and a user-centric design philosophy, Lectoro is positioned to enhance data accuracy, optimise administrative procedures, and promote an engaged and accountable culture within the academic community.

Moving forward, continued cooperation, communication, and adaptability to changing requirements and technology breakthroughs will be necessary for the effective implementation of Lectoro. Lectoro will continue to be at the forefront of attendance management systems thanks to ongoing feedback loops within our team, usability testing, and system improvements. This will enable UOW to embrace innovation and maximise resource utilisation for the benefit of all stakeholders.

This design report, captures the process we will be going through to turn a vision into reality, representing the combined commitment, knowledge, and ingenuity of all those involved. Lectoro is a monument to the ability of design thinking, group problem-solving, and technology-driven innovation to influence the direction of education.

REFERENCES

Requirement’s Analysis:

Power BI - Data Visualization | Microsoft Power Platform 2023, Microsoft.com, viewed 3 May 2024, <https://www.microsoft.com/en-us/power-platform/products/power-bi/>

Tableau Desktop 2024, Tableau, viewed 3 May 2024, <https://www.tableau.com/products/desktop>

Canvas by Instructure | World’s #1 Teaching and Learning Software 2021, Instructure, viewed 3 May 2024, <https://www.instructure.com/canvas>.

Target Market:

Jibble 2024, Jibble, viewed 3 May 2024, <https://www.jibble.io/face-recognition-attendance-system>.

Samson Kiarie 2024, *Best Face Recognition Attendance System in 2024*, Timeero.com, Timeero, viewed 3 May 2024, <https://timeero.com/post/best-face-recognition-attendance-system>.

Facial Recognition Feature | Software Uses & Features | Buddy Punch 2024, Buddy Punch, viewed 3 May 2024, <https://buddypunch.com/time-clock-software/features/facial-recognition/>.

Tools:

Microsoft 2021, *Visual Studio Code*, Visualstudio.com, Microsoft, viewed 3 May 2024, <https://code.visualstudio.com/>.

Microsoft 2021, *Visual Studio Code*, Visualstudio.com, Microsoft, viewed 3 May 2024, <https://code.visualstudio.com/docs>

JetBrains 2024, *IntelliJ IDEA*, JetBrains, JetBrains, viewed 3 May 2024, <https://www.jetbrains.com/idea/>.

Repositories documentation - GitHub Docs 2024, GitHub Docs, viewed 3 May 2024, <https://docs.github.com/en/repositories>.

Cloud Computing Services - Amazon Web Services (AWS) 2024, Amazon Web Services, Inc., viewed 3 May 2024, <https://aws.amazon.com/>.

Amazon S3 - Cloud Object Storage - AWS 2024, Amazon Web Services, Inc., viewed 3 May 2024, <https://aws.amazon.com/s3/>

Back-end Design:

The Backend With Java and Spring Boot 2020, Milanwittpohl.com, viewed 3 May 2024, <https://www.milanwittpohl.com/projects/tutorials/full-stack-web-app/the-backend-with-java-and-spring>.

Spring Modulith 2024, Spring Modulith, viewed 3 May 2024, <https://spring.io/projects/spring-modulith>.

Introducing Spring Modulith 2022, Introducing Spring Modulith, viewed 3 May 2024, <https://spring.io/blog/2022/10/21/introducing-spring-modulith>.

Spring | Why Spring 2024, Why Spring, viewed 3 May 2024, <https://spring.io/why-spring>.

Spring | Microservices 2024, Microservices, viewed 3 May 2024, <https://spring.io/microservices>.

MyExamCloud 2023, *Integrating Python and Java: A Guide for Developers*, Linkedin.com, viewed 3 May 2024, <https://www.linkedin.com/pulse/integrating-python-java-guide-developers-myexamcloud-7g5bc>.

Development Environment:

7 Best Computer Vision Libraries in Python 2024, ProjectPro, viewed 3 May 2024, <https://www.projectpro.io/article/computer-vision-libraries/772>.

OpenCV 2024, OpenCV, viewed 3 May 2024, <https://opencv.org/>.

NumPy 2023, Numpy.org, viewed 3 May 2024, <https://numpy.org/>.

face-recognition 2020, PyPI, viewed 3 May 2024, <https://pypi.org/project/face-recognition/>.

scikit-learn: machine learning in Python 2024, Scikit-learn.org, viewed 3 May 2024, <https://scikit-learn.org/stable/>.

An introduction to seaborn 2024, Pydata.org, viewed 3 May 2024, <https://seaborn.pydata.org/tutorial/introduction.html>.

Microsoft 2021, *Visual Studio Code*, Visualstudio.com, Microsoft, viewed 3 May 2024, <https://code.visualstudio.com/>.

Microsoft 2021, *Visual Studio Code*, Visualstudio.com, Microsoft, viewed 3 May 2024, <https://code.visualstudio.com/docs/languages/python>.

JetBrains 2024, *IntelliJ IDEA*, JetBrains, JetBrains, viewed 3 May 2024, <https://www.jetbrains.com/idea/>.

IntelliJ IDEA Help 2024, IntelliJ IDEA Help, viewed 3 May 2024, <https://www.jetbrains.com/help/idea/spring-boot.html>.

JetBrains 2024, *PyCharm*, JetBrains, JetBrains, viewed 3 May 2024, <https://www.jetbrains.com/pycharm/>.

MongoDB: The Developer Data Platform 2024, MongoDB, viewed 3 May 2024, <https://www.mongodb.com/>.

Schemaless Database 2024, MongoDB, viewed 3 May 2024, <https://www.mongodb.com/resources/basics/unstructured-data/schemaless#:~:text=No%20data%20truncation,to%20match%20the%20current%20schema.>.

Arti Technologies 2024, *Process of Storing Images in MongoDB - MongoDB Tutorial 2024 Latest Version - Medium*, Medium, MongoDB Tutorial 2024 Latest Version, viewed 3 May 2024, <https://medium.com/mongodb-tutorial/process-of-storing-images-in-mongodb-80c0f928cbd3>.

Deployment Environment:

Cloud Computing Services - Amazon Web Services (AWS) 2024, Amazon Web Services, Inc., viewed 3 May 2024, <https://aws.amazon.com/>.

Front-end Design:

What is RESTful API? - AWS 2024, Amazon Web Services, Inc., viewed 3 May 2024, <https://aws.amazon.com/what-is/restful-api/#:~:text=RESTful%20API%20is%20an%20interface,applications%20to%20perform%20various%20tasks.>.

Tailwind 2020, Tailwindcss.com, Tailwind CSS, viewed 3 May 2024, <https://tailwindcss.com/>.

Themesberg 2019, *Flowbite - Tailwind CSS component library*, Flowbite.com, viewed 3 May 2024, <https://flowbite.com/docs/getting-started/introduction/>.