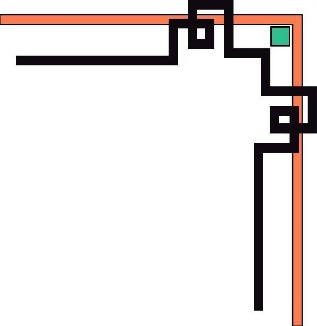
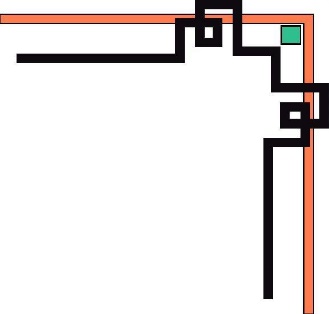
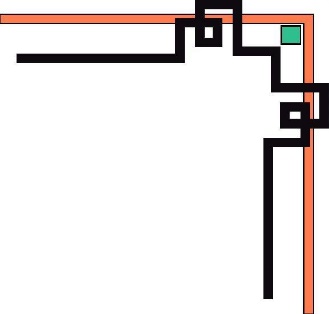
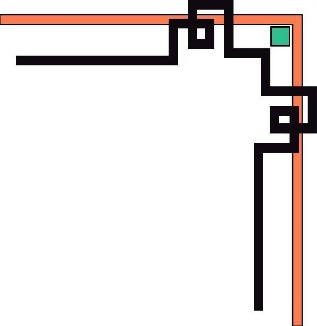
**MINISTRY OF EDUCATION AND TRAINING**



**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

 --------------------

**CAPSTONE PROJECT**

**BUILDING A WEBSITE FOR STUDENT ATTENDANCE**

**INSTRUCTOR: Nguyen Tran Thi Van MSc.**

STUDENTS:

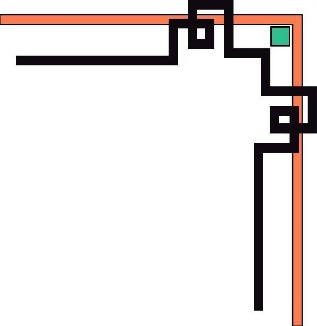
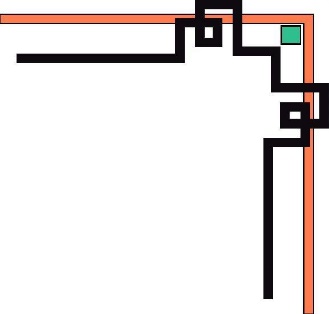
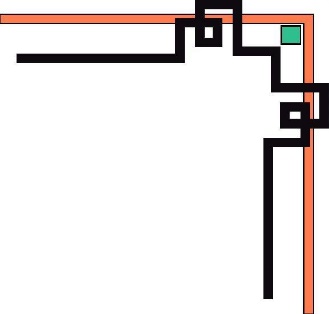
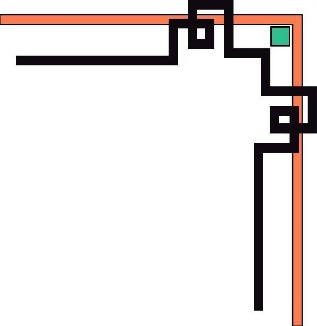
Nguyễn Tấn Thành ID Student: 18110046

Hoàng Ngọc Bảo Trân ID Student: 18110057

Nguyễn Hoàng Vũ ID Student: 18110398

**Hồ Chí Minh City, February, 2023**

**MINISTRY OF EDUCATION AND TRAINING**



**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

 --------------------

**CAPSTONE PROJECT**

**BUILDING A WEBSITE FOR STUDENT ATTENDANCE**

**INSTRUCTOR: Nguyen Tran Thi Van MSc.**

STUDENTS:

Nguyễn Tấn Thành ID Student: 18110046

Hoàng Ngọc Bảo Trân ID Student: 18110057

Nguyễn Hoàng Vũ ID Student: 18110398

**Hồ Chí Minh City, February, 2023**



**SOCIALIST REPUBLIC OF VIETNAM**

**Independence – Freedom - Happiness**

--------

*Ho Chi Minh City, September 15th , 2022*

## **PROJECT ASSIGNMENT**

Student Name: Nguyễn Tấn Thành ID: 18110046

Student Name: Hoàng Ngọc Bảo Trân ID: 18110057

Student Name: Nguyễn Hoàng Vũ ID: 18110398

Specialization: Capstone Project.

Project title: Building a Website For Student Attendance.

**Implementation of Content**

*Theory:*

* Research and learn React.JS.
* Research and learn Node.JS.
* Research and learn MongoDB.
* Research and learn Express JS.

*Practice:*

Build an Website:

* For student: login/out, registration confirmation email, view profile, edit profile, view project, check attendance, view history attendance, manage personal information, send request, search request, view request, recive task, submit task, view task, view project, join project.
* For teacher: login/out, registration confirmation email, manage project, view project, create project, manage students, check attendance, view history attendance, manage student information, manage personal information, send task, view task.

**Processing time:** 15 weeks

Student's signature: *Nguyễn Tấn Thành 18110046*

Student's signature: *Hoàng Ngọc Bảo Trân 18110057*

Student's signature: *Nguyễn Hoàng Vũ 18110398*

**CHAIR OF THE PROGRAM INSTRUCTOR**

*(Sign with full name) (Sign with full name)*

 **SOCIALIST REPUBLIC OF VIETNAM**

**Independence – Freedom - Happiness**

--------

*Ho Chi Minh City, February, 2023*

**COMMENTARY OF INSTRUCTOR**

Student Name: Nguyễn Tấn Thành ID: 18110046

Student Name: Hoàng Ngọc Bảo Trân ID: 18110057

Student Name: Nguyễn Hoàng Vũ ID: 18110398

Major: Capstone Project

Project title: Building a Website For Student Attendance.

Name of Instructor: Nguyen Tran Thi Van MSc.

**EVALUATION**

1. Content and workload of the project:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

2. Strengths:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

3. Weaknesses:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

4. Approval for oral defense? *(Approved or denied)*

.......................................................................................................................................

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.......................................................................................................................................

6.  Mark: ………………. (*In words*: ...........................................................................)

**INSTRUCTOR**

*(Sign with the full name)*

 **SOCIALIST REPUBLIC OF VIETNAM**

**Independence – Freedom - Happiness**

--------

*Ho Chi Minh City, February, 2023*

**COMMENTARY OF REVIEWER**

Student Name: Nguyễn Tấn Thành ID: 18110046

Student Name: Hoàng Ngọc Bảo Trân ID: 18110057

Student Name: Nguyễn Hoàng Vũ ID: 18110398

Major: Capstone Project.

Project title: Building a Website For Student Attendance.

Name of Reviewer: Nguyen Thien Bao PhD.

**EVALUATION**

1. Content and workload of the project:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

2. Strengths:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

3. Weaknesses:

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

4. Approval for oral defense? *(Approved or denied)*

.......................................................................................................................................

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.......................................................................................................................................

6.  Mark: ………………. (*In words*: ...........................................................................)

**REVIEWER**

*(Sign with the full name)*

## **ACKNOWLEDGEMENT**

Thank our instructor – Nguyen Tran Thi Van MSc. for amazing lectures and discussions! We enjoyed this class, we're excited to come to class every lesson, learned a lot, and benefited from your course. Thanks for facilitating such a positive learning environment.

Thank you so much for caring about both what you’re teaching and how you teach it for this semester. For your patience and for creating an environment where we aren’t only allowed to succeed but also to fail. For caring for the students and training us. For allowing us to question without judging and for having a growth mindset with us. Students succeed when students feel the freedom to image and trust they give. You are a very good supervisor. You continue to impress us more and more with your knowledge, creativity, and engaging coursework.

However, our limited knowledge and experience in technology and social work may prevent us from developing software that is perfect for everyone. We eagerly await your feedback in order to improve the software even further.

Hồ Chí Minh City, 22 February, 2023.

Student's signature: Nguyễn Tấn Thành

Student's signature: Hoàng Ngọc Bảo Trân

Student's signature: Nguyễn Hoàng Vũ

## **TABLE OF CONTENTS**

[**CHAPTER 1: INTRODUCTION** 1](#_Toc127930547)

[**1.** **Reason to select this topic** 1](#_Toc127930548)

[**2.** **The goal of the project** 1](#_Toc127930549)

[**3.** **Scope** 1](#_Toc127930550)

[**CHAPTER 2: THEORY BASIS** 2](#_Toc127930551)

[**2.1. MERN Stack** 2](#_Toc127930552)

[2.1.1. ReactJS 2](#_Toc127930553)

[2.1.2. NodeJS 4](#_Toc127930554)

[2.1.3. ExpressJS 7](#_Toc127930555)

[2.1.4. MongoDB 7](#_Toc127930556)

[**CHAPTER 3: SYSTEM REQUIREMENTS AND MODELING** 9](#_Toc127930557)

[**3.1.** **Related works:** 9](#_Toc127930558)

[**3.2.** **System requirements:** 9](#_Toc127930559)

[3.2.1. Functional Requirements 9](#_Toc127930560)

[3.2.2. Non-functional Requirements 14](#_Toc127930561)

[**3.3.** **Modeling** 16](#_Toc127930562)

[3.3.1. Usecases modeling 16](#_Toc127930563)

[3.3.2. Usecases diagram 19](#_Toc127930564)

[**CHAPTER 4: SYSTEM DESIGN** 34](#_Toc127930565)

[**4.1. Database Design** 34](#_Toc127930566)

[**4.2. UI Design** 39](#_Toc127930567)

[4.2.1. List of screens and conversion diagrams 39](#_Toc127930568)

[4.2.2. Detailed description of the screens 39](#_Toc127930569)

[**4.3. Sequence Diagrams** 48](#_Toc127930570)

[**CHAPTER 5: IMPLEMENTATION AND TESTING** 58](#_Toc127930571)

[**5.1. Implementation** 58](#_Toc127930572)

[5.1.1. Apply the MERN stack 58](#_Toc127930573)

[**5.2. Testing** 61](#_Toc127930574)

[5.2.1. Test Strategy 61](#_Toc127930575)

[5.2.2. Test Environment 63](#_Toc127930576)

[5.2.3. Testing method and techniques 63](#_Toc127930577)

[5.2.4. Black box testing and bug report 67](#_Toc127930578)

[**CHAPTER 6: CONCLUSION** 73](#_Toc127930579)

[**6.1.** **Achievements** 73](#_Toc127930580)

[6.1.1. Theoretically achievements 73](#_Toc127930581)

[6.1.2. Realistic results 73](#_Toc127930582)

[**6.2.** **Limit** 73](#_Toc127930583)

[**6.3.** **Directions of future development** 74](#_Toc127930584)

[**REFERENCES** 75](#_Toc127930585)

**LIST OF FIGURES**

[Figure 1: User usecase diagram 22](#_Toc127933387)

[Figure 2: List of screens diagram 43](#_Toc127933388)

[Figure 3: Landing page design 44](#_Toc127933389)

[Figure 4: Sign in form design 44](#_Toc127933390)

[Figure 5: Sign up form design 45](#_Toc127933391)

[Figure 6: Design personal information page 46](#_Toc127933392)

[Figure 7: Homepage information 46](#_Toc127933393)

[Figure 8: All students page 48](#_Toc127933394)

[Figure 9: Project manage page 49](#_Toc127933395)

[Figure 10: Create project popup 50](#_Toc127933396)

[Figure 11: Add student popup 51](#_Toc127933397)

[Figure 12: Login diagram 52](#_Toc127933398)

[Figure 13: Check attendance diagram 52](#_Toc127933399)

[Figure 14: Send absent request diagram 53](#_Toc127933400)

[Figure 15: Search project diagram 53](#_Toc127933401)

[Figure 16: View profile diagram 54](#_Toc127933402)

[Figure 17: View teacher profile diagram 54](#_Toc127933403)

[Figure 18: View history attendance diagram 55](#_Toc127933404)

[Figure 19: View project information diagram 55](#_Toc127933405)

[Figure 20: Add student to project diagram 56](#_Toc127933406)

[Figure 21: View students diagram 56](#_Toc127933407)

[Figure 22: Delete student diagram 57](#_Toc127933408)

[Figure 23: Student profile diagram 57](#_Toc127933409)

[Figure 24: Project management diagram 58](#_Toc127933410)

[Figure 25: Project management diagram 58](#_Toc127933411)

[Figure 26: Project management diagram 59](#_Toc127933412)

[Figure 27: Attendance management diagram 59](#_Toc127933413)

[Figure 28: Project management diagram 60](#_Toc127933414)

[Figure 29: Front-end work 61](#_Toc127933415)

[Figure 30: Structure ReactJS in project 62](#_Toc127933416)

[Figure 31: MVC pattern in NodeJS 63](#_Toc127933417)

[Figure 32: REST API 63](#_Toc127933418)

## **LIST OF TABLES**

[Table 1: Teacher role in page 11](#_Toc127932348)

[Table 2: Student role in page 14](#_Toc127932349)

[Table 3: User related in non-functional requirements 17](#_Toc127932350)

[Table 4: Staff-related in non-requirement function 17](#_Toc127932351)

[Table 5: Usecases actor and function 18](#_Toc127932352)

[Table 6: Actor description 19](#_Toc127932353)

[Table 7: Function description 20](#_Toc127932354)

[Table 8: Login usecase specification 22](#_Toc127932355)

[Table 9: Register usecase specification 24](#_Toc127932356)

[Table 10: View project information usecase specification 25](#_Toc127932357)

[Table 11: Search project usecase specification 26](#_Toc127932358)

[Table 12: View history attendance usecase specification 27](#_Toc127932359)

[Table 13: Send a request usecase specification 28](#_Toc127932360)

[Table 14: Check attendance usecase specification 30](#_Toc127932361)

[Table 15: View student/ request/ project usecase specification 31](#_Toc127932362)

[Table 16: Edit student/ request/ project information usecase specification 32](#_Toc127932363)

[Table 17: Delete student/ request/ project specification 33](#_Toc127932364)

[Table 18: Add student/ request/ project specification 34](#_Toc127932365)

[Table 19: Acceptance request usecase specification 36](#_Toc127932366)

[Table 20: User database 38](#_Toc127932367)

[Table 21: Project database 39](#_Toc127932368)

[Table 22: Request database 40](#_Toc127932369)

[Table 23: Task database 41](#_Toc127932370)

[Table 24: Student join database 41](#_Toc127932371)

[Table 25: studentSchedules database 42](#_Toc127932372)

[Table 26: Test Deliverables 65](#_Toc127932373)

# **CHAPTER 1: INTRODUCTION**

## **Reason to select this topic**

Nowaday, a large part of the students are giddgy or getting a job soon. Therefore they not pay attention to go to school. Attendance is also a way to encourage students to come to school and study seriously.

## **The goal of the project**

Give users a good experience when coming to our web. Our team wanted to make a website for teachers to strictly manage their students' attendance. In addition, students can also easily check their attendance.

Research mission

* Clarifying the theoretical basis of the topic.
* Survey the actual situation of the current system.
* Design the website according to the structure and color scheme of the eCommerce website.
* The functionality of the website is divided into different modules for ease of use management.

## **Scope**

Learn around ReactJS, NodeJS, MongoDB,… to build an Student Attendance website.

# **CHAPTER 2: THEORY BASIS**

## **2.1. MERN Stack**

The technology we chose to develop this project is the MERN tech stack (MongoDB, Express, ReactJS, NodeJS). Because it is one of the most modern web development technologies, it supports the development of a website both backend and frontend. MERN stack makes it possible for developers to build a website in less time suitable for agile development projects. Learning these technologies is also quite easy and does not take much time to apply. In addition, the support community and resources are very large, so the application of these technologies to understanding and developing web applications is completely reasonable and feasible.

### 2.1.1. ReactJS

1. Theory

ReactJS is a JavaScript-based UI development library. Facebook and an open-source developer community run it. Although ReactJS is a library rather than a language, it is widely used in web development. The library first appeared in May 2013 and is now one of the most commonly used frontend libraries for web development.

ReactJS offers graceful solutions to some of front-end programming’s most persistent issues, allowing you to build dynamic and interactive web apps with ease. It’s fast, scalable, flexible, powerful, and has a robust developer community that’s rapidly growing. There’s never been a better time to learn React.

1. Main concept

React is designed around the concept of reusable components. You define small components and you put them together to form bigger components.

All components small or big are reusable, even across different projects.

Components in ReactJS play the same role as functions in javascript, but they work independently and are responsible for returning HTML components through the render function.

There are 2 types of components: Function components and Class components

* Function Component: in ReactJS, the easiest way to define a component is to write a JavaScript function, which returns the HTML response:
* Class Component: we can also use ES6 Class to define a component, this is called the Class component in ReactJS

1. Outstanding features

You can completely use ReactJS to write apps directly in JavaScript. Turn complex UI components into standalone components. Pass customized data to a specific UI component. Allows you to change the state for many child components and not affect the parent component even while in the Stateful state. The ability to determine exactly when to re-render as well as when to remove the DOM element.

* **User-friendly interface:** using ReactJS, users can create attractive, fast, and efficient web applications with minimal coding. Users will feel the optimization of ReactJS in individual components. Instead of having to work on the entire website application, when using ReactJS, developers can break down the UI structure into simpler components. With the level of a JavaScript programmer, using ReactJS, you can easily develop web-based applications.
* **Support Reusable Component:** not every developer can design effective components. Reusable components allow developers to make extensive use of pre-optimized features. Thanks to ReactJS, developers can completely reuse components that have been developed into other applications with the same functionality.
* **Write components more easily and faster**: to write React components, users will use JSX. This is a combination of JavaScript and HTML. While not the most popular syntax extension, JSX is appreciated by users when developing special components or large applications. Using JSX, users can extend the syntax with a wide range of options easily. The process of writing site structure is also clearer.
* **Better performance with Virtual DOM:** web-based application projects are often prone to trouble in the DOM process. With the advantage of using Virtual-DOM, ReactJS will help users avoid this problem. Besides, ReactJS also allows users to build Virtual DOM and host them in memory. This advantage will help virtual change as soon as the actual DOM changes. Continuous updates help apps avoid interruptions. Users use it continuously, and the website is stable, helping to maintain and increase revenue.
* **SEO Friendly:** this is the wish of all businesses in today's exciting e-commerce era. The more the user interface is accessible to different search engines, the higher the ability to reach customers. This is one of the factors that directly affect the growth of sales through the website. In fact, not all JavaScript frameworks are SEO-friendly. Fortunately, ReactJS was born as a solution to this problem. React allows you to create user interfaces that can be accessed on different search engines. However, ReactJS is just a JavaScript library. ReactJS cannot do everything by itself, replacing the entire SEO, SALE, or Developer team. Users can combine ReactJS with additional libraries needed for management, routing, and interaction purposes.

### 2.1.2. NodeJS

1. Theory

NodeJS is an open-source development platform for executing JavaScript code server-side based platform for server-side programming built on Chrome's V8 JavaScript engine, which means that the engine that compiles JavaScript in a web Browser it's the same engine that runs Node.js in its core, and this is what makes it so fast, especially for a web application. Node is useful for developing applications that require a persistent connection from the browser to the server and is often used for real-time applications such as chat, news feeds, and web push notifications.

1. Architecture

Node.js architecture has 3 key elements that work together as an event processing factory. So, to make it easier to understand how these elements interact with each other, I will give an example of a REST API developed in Node.js:

* **Event Queue**: as soon as these requests reach the application, they go to, which is a queue where all the events that occur in the application go first, and where they await to be sent to be processed in the main thread called Event Loop.
* **Event loop**: When a request (Blocking Operation) enters the Event Loop, which is a single thread platform that runs the V8 Engine in its core to compile JavaScript, it’s delegated to the Thread Pool platform to be processed in the background. So, with this architectural flow, the main thread is available again to handle other events.
* **Thread Pool**: in the Thread Pool, which is a multi-thread platform that runs a library called the library and has C++ in its core, the request (Blocking Operation) is processed asynchronously in the background until it’s completed and ready to be returned.

1. Outstanding features

Node.js is an extremely powerful server-side platform to develop modern, reliable, and scalable web applications, trusted by global companies such as Netflix, Uber, LinkedIn, and PayPal.

Due to its single thread characteristics combined with a multi-thread platform that runs in the background, its event-driven architecture is perfect for applications with intense I/O (Inputs and Outputs), making it serve more clients with less hardware, which means more performance with fewer costs.

* **Very Fast**: Being built on Google Chrome's V8 JavaScript Engine, the Node.js library is very fast in code execution.
* **Non-Blocking I/O:** The non-blocking method receives input and returns the output asynchronously. In other words, it makes it possible for a server to receive many requests without blocking the application while the response is being processed in the background.
* **Single Threaded but Highly Scalable**: Node.js uses a single-threaded model with event looping. The event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single-threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server. Node.js can deal with many events at the same time with its single thread characteristic that delegates the asynchronous operations to a multi-thread platform, which means that just one thread can handle inputs and outputs. This feature makes it lightweight, scalable, and efficient by consuming low memory and infrastructure capacity to process a huge number of requests.
* **Event-Driven**: Node.js is an event-driven technology, which means that the control flow of this server-side platform is driven by the occurrence of events. So, at the moment that a Node application starts, an event listener called Event Loop begins to wait for events and doesn't stop until the application is shut down.
* **Node Package Manager**: Node Package Manager it’s the world's largest free and open-source library of functionalities and can be easily imported and used in any Node application. All this means that, regardless of the tools that an application needs, it will be found at Node Package Manager.
* **No Buffering**: Node.js applications never buffer any data, which dramatically reduces the processing time of uploading files, such as videos or audios. These applications simply output the data in chunks. For example, a user can watch videos without any interruption
* **Scalable**: Scalability is a core Node characteristic, due to a high capacity of handling large amounts of requests asynchronously with low infrastructure by its architecture that operates on a single thread combined with a multi-thread platform, allowing it receives thousands of simultaneous events.
* License − Node.js is released under the [MIT license](https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE)

### 2.1.3. ExpressJS

Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

### 2.1.4. MongoDB

1. Theory

MongoDB is a document-oriented NoSQL database used for high-volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables

1. Features
2. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
3. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
4. The rows (or documents as called in MongoDB) don’t need to have a schema defined beforehand. Instead, the fields can be created on the fly.
5. The data model available within MongoDB allows you to represent hierarchical relationships, store arrays, and other more complex structures more easily.
6. Scalability –MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with millions of documents within the database.

# **CHAPTER 3:** **SYSTEM REQUIREMENTS AND MODELING**

## **Related works:**

We found that most of the learning and the internet are interrelated such as online materials, online learning and even online tests but online attendance is not available or well known.Therefore, we decided build an online attendance website to make attendance in class is faster, more convenient and more accurate.

## **System requirements:**

### Functional Requirements

* Business functional requirements:
* **Storage**: All information about the
* Users: email, username, password, firstName, lastName, dateOfBirth, phoneNumber, address, role.
* Project: projectName, projectDecription, startDate, endDate, learnDate.
* Attendance: projectID, status, indexOfTime.
* Task: id, projectID, submitted, files, title, description, startTime, endTime, createdBy
* Request: id, projectID, datetime, reason, type.
* **Searching:**
* Search student’s information.
* Search project.
* Search request.
* Search task.
* Search attendance.
* **Calculation:**
* Percentage of progress.
* Total student.
* Total subject.
* Total task.
  + **Reporting:**
* History attendance.
* Today activities.

*Table 1: Teacher role in page*

**Division (user): Teacher ID: Tc**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OD** | **Function** | | **Type** | **Constraint/ Formula Code** | **Form code** | **Note** |
| 1 | Sign in | Sign in | | Login function needs:   * Username * Password |  |  |
| 2 | Search student | Manage student | | Search student function is based on ID or student name. |  |  |
| 3 | Add student | Manage student | | Add student function needs:   * Email * First name * Last name * Student id * Class * Major * Gender * Date of birth * Address |  |  |
| 4 | Edit student | Manage student | | Edit student function needs:   * Email * First name * Last name * Student id * Class * Major * Gender * Date of birth * Address |  |  |
| 5 | Add project | Manage project | | Add project function needs:   * Project name * Start day * End day * Max join * Total lesson * Attendance after * Description * Learn day |  |  |
| 6 | Edit project | Manage project | | Edit project function needs:   * Project name * Start day * End day * Max join * Total lesson * Attendance after * Description * Learn day |  |  |
| 7 | Delete project | Manage project | | Delete project function needs: projectIds |  |  |
| 8 | Add request | Manage request | | Delete request function needs:   * Request id * Reason * Date |  |  |
| 9 | Accept request | Manage request | | Make request function needs:   * Status * requestId |  |  |
| 10 | Edit request | Manage request | | Make request function needs:   * requestId * Status |  |  |
| 11 | Send task | Manage task | | Send task function needs:   * Title * Decription * File * Start day * End day |  |  |
| 12 | View task | Manage task | | Manage task function needs:  Submitted list |  |  |
| 13 | Edit information | Information | | Edit information function needs:   * Email * Username * First name * Last name * Password * Repeat password * Phone * Address * Date of birth |  |  |
| 14 | Sign up | Sign up | | Login function needs:   * Email * Username * First name * Last name * Password * Repeat password * Phone * Address * Date of birth * Valida email |  |  |

*Table 2: Student role in page*

**Division (user): Students ID: std**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OD** | **Function** | **Type** | **Constraint/ Formula Code** | **Form code** | **Note** |
| 1 | Sign in | Sign in | Login function needs:   * Username * Password |  |  |
| 2 | Sign up | Sign up | Register function needs:   * Register by filling the form * Email * Username * First name * Last name * Password * Repeat password * Phone * Address * Date of birth |  |  |
| 3 | Check attendance | Attendance | Check attendance function is based on :   * Project name * Attendance after |  |  |
| 4 | View history attendance | Attendance | View history function needs:   * Order * Date * Status |  |  |
| 5 | Search history attendance | Attendance | Search history function needs:   * Date * Month * Year |  |  |
| 6 | Add request | Request | Add request function needs:   * Project name * Project ID * Reason * Date |  |  |
| 7 | Search request | Request | Search request function needs:   * Date * Request type * Request status |  |  |
| 8 | Search task | Task | Search task function needs:   * Project name |  |  |
| 9 | Join project | Project | Join project function needs:   * Project name * Project ID * Reason * Date |  |  |
| 10 | Edit information | Information | Edit information function needs:   * Email * Username * First name * Last name * Password * Repeat password * Phone * Address * Date of birth |  |  |

* **System function requirements:**
* **Environment:** The system operates on a Website, student and teacher can access through provided accounts:
* **Decentralization:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Contents** | **Description** | **Notes** |
| 1 | Role teacher | * Administrator: allow all roles. * Student are allowed to add request, check attendance, edit personal profile. |  |

### Non-functional Requirements

* **User related:**

*Table 3: User related in non-functional requirements*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Contents** | **Criteria** | **Description** | **Note** |
| 1 | The software can be updated and modified to adapt to the trend of users as well as the requirements of managing the owners. | Resilience |  |  |
| 2 | Graphic User-Interface:   * User-friendly. * Easy manipulation. * Feature buttons are laid out simple, convenient, and visible. * Functional windows can interact with each other, providing users with efficiency. | Usability |  |  |
| 3 | * Database is kept secure and easy to access * The system operates stable, reliable, and instant response. | Efficiency |  |  |

* **Staff-related:**

*Table 4: Staff-related in non-requirement function*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Contents** | **Criteria** | **Description** | **Note** |
| 1 | * The module, source code, and other proportions of the system can be reused for later projects. * The source code can be recycled and applied for other programs without changing the cores. | Reusability |  |  |

* **Quality**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Contents** | **Criteria** | **Description** | **Note** |
| 1 | The search product function is convenient, natural, and visible. | Convenience | Support the keyword-like searching. |  |
| 2 | Processing time is acceptable | Effectiveness | Optimize for printing an order processing is less than 2 seconds. |  |

## **Modeling**

### Usecases modeling

#### Define usecases actor and function

*Table 5: Usecases actor and function*

|  |  |
| --- | --- |
| **Actors** | **Features** |
| **Teacher** | * Login * Sign up * Add student * Delete student * Edit student information * View student information * Search student by name * Contact student by Email * Search attendance history * Manage attendance * Add request * Edit request * View request * Accept request * Add project * Edit project day * Edit student in project * Search project * View project * Delete project * Create task * Edit task * View detail task * Logout |

#### Actor and function description

#### Actor description

*Table 6: Actor description*

|  |  |
| --- | --- |
| Actor | Features |
| **Student** | * Login * Sign up * View teacher profile * View project profile * Edit profile * View history attendance * Send absent request * Create request for join project * View history request * Search request * Check attendance * View history attendance * View task * Submit task * Join project * View project * Edit information * Log out |

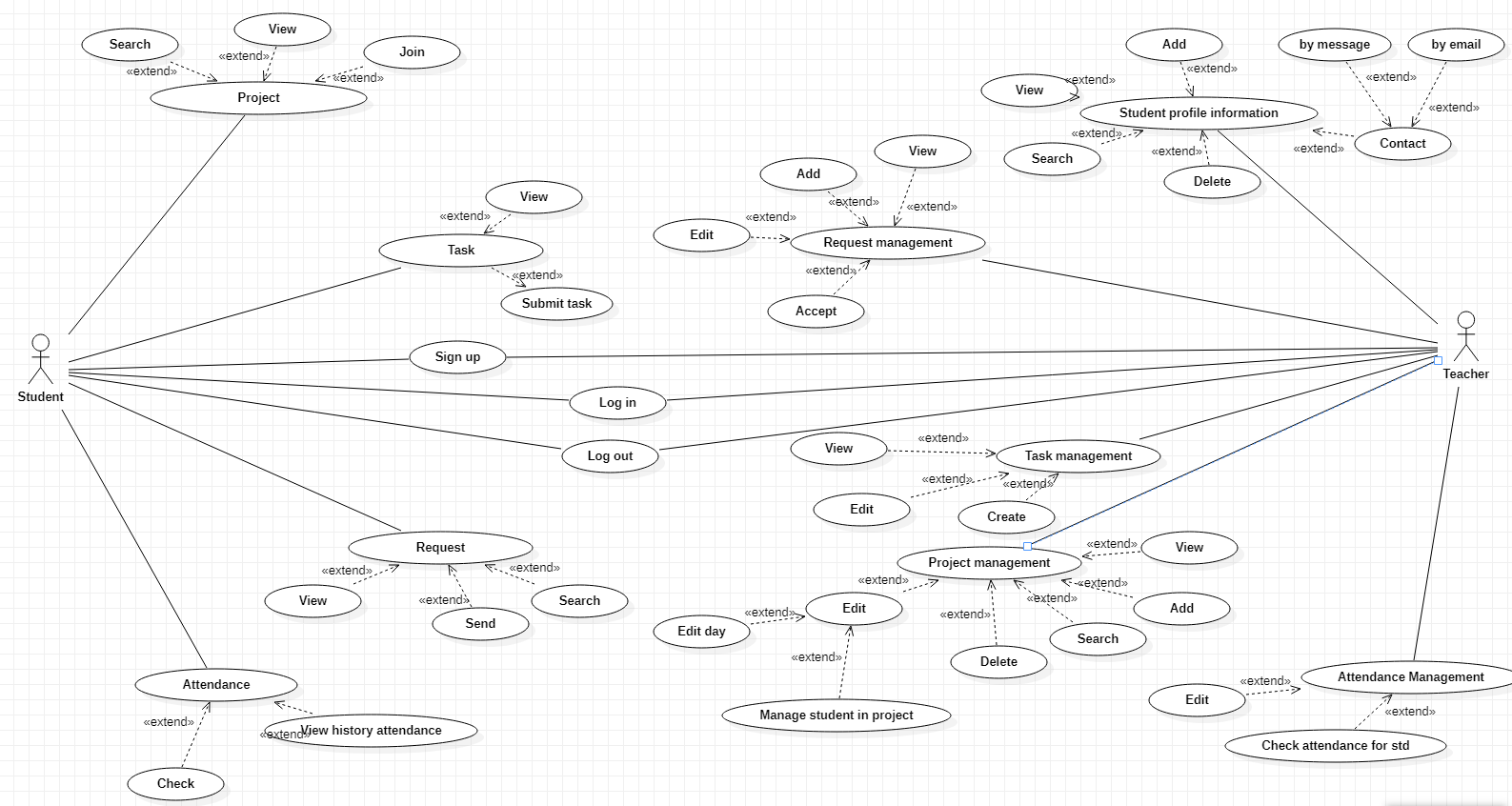
#### Function description

*Table 7: Function description*

|  |  |  |
| --- | --- | --- |
| Number | Feature | Description |
| 1 | Login | Allow users to log in to the system:   * Enter the registered username and password. |
| 2 | Register | Allow users to create an account to log into the system. |
| 3 | Search project | Allow users to search for project. |
| 4 | View project | Allow users to find project information. |
| 5 | View history attendance | Allow users to view history of attendances. |
| 6 | Send request | Allow user can send a request based on the user’s reason to teacher. |
| 7 | Check attendance | Allow users to check attendance in today class. |
| 8 | Contact | Allow users to text online or send mail with each others. |
| 9 | Add student | Allow users to add student in to system. |
| 10 | Edit student | Allow users to edit student in system. |
| 11 | Delete student | Allow users to delete student in system. |
| 12 | View student | Allow users to find student information. |
| 13 | Search student | Allow users to search for student. |
| 14 | Accept request | Allow user to accept a request from student. |
| 15 | Edit project | Allow users to edit day and student in project. |
| 16 | Check attendance for student | Allow users to check and edit attendance for student in class. |
| 17 | Create task | Allow teacher to add task for student in project. |
| 18 | View task | Allow user to view task information. |
| 19 | Log out | Allow user can log out of account on page |

### Usecase diagram

#### Usecase diagram



*Figure 1: User usecase diagram*

#### Usecase specification

#### Log in

*Table 8: Login usecase specification*

|  |  |
| --- | --- |
| **Name** | Login |
| **Brief Description** | The way users log into the site |
| **Actor(s)** | All users can login into the system |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user wants to access the system   1. User select “Login” function on the system 2. The system displays the “Login” page 3. User enter the correct username and password then click "Login". 4. The system will return the user to the Homepage. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| Enter the wrong login information | 1. If the user enters an invalid password:  1.1. The system will ask the user to re-enter the password.   * The system determined the password is incorrect.   2. If the user enters an invalid username:   * Error system. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | The user already has an account on the system |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Login successfully, the user accesses the system. |
| Failure | Login failed, the user cannot log in. |
| **Extension Points** | |
| None | |

#### Register

*Table 9: Register usecase specification*

|  |  |
| --- | --- |
| **Name** | Register |
| **Brief Description** | The way user creates a personal account |
| **Actor(s)** | All users |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user chooses the function to register   1. The user chooses the "Register" function in the system. 2. The system displays the "Register’’ page with information about username, password, email. 3. The user fills out the above information and presses the "submit” 4. The notification system has successfully registered. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| Fill out the information missing | 1. The system requires re-fill and full information on the "Register" page. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | User successfully connected to the system |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Register successfully, the user creates an account. |
| Failure | Register failed, the user creates account failure. |
| **Extension Points** | |
| None | |

#### View project information

*Table 10: View project information usecase specification*

|  |  |
| --- | --- |
| **Name** | View *project* |
| **Brief Description** | The way student and teacher view all their project. |
| **Actor(s)** | Student, teacher. |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user chooses the function to View the project information   1. User access Project. 2. The user clicks the "More Information" button. 3. The system shows the information about the project. | |
| **Alternate Flows** | |
| **Title** | **Description** |
|  |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | User successfully accessed the “Project more info” |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | user view project successfully. |
| Failure | user view project failure. |
| **Extension Points** | |
| None | |

#### Search project

*Table 11: Search project usecase specification*

|  |  |
| --- | --- |
| **Name** | Search project information |
| **Brief Description** | The way all users project information |
| **Actor(s)** | All users |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user chooses the function to search project information   1. The user chooses the "Search project " function in the system. 2. The user enters information and presses the "Search" button. 3. The system displays project information according to the keyword entered. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| The user enters a keyword that is not in the database | 1. Notification system doesn't have the kind of project that user looking for. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Search successful, user search information successfully. |
| Failure | The search failed, user search information failure. |
| **Extension Points** | |
| None | |

#### View history attendance

*Table 12: View history attendance usecase specification*

|  |  |
| --- | --- |
| **Name** | View history attendance |
| **Brief Description** | The way users view all history attendance by search day, month or year. |
| **Actor(s)** | All users |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user go to Attendance page and start search by their keyword   1. User access Attendance site 2. Choose what they want search by (Day, Month or Year) 3. The user types keyword in search bar. 4. The system shows call API to third-party to get the results | |
| **Alternate Flows** | |
| **Title** | **Description** |
|  |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | User successfully accessed the “Attendance”  The connection between the system's server and the third party has not been broken |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Call API to third-party successfully. |
| Failure | Fail integrate with third-party. |
| **Extension Points** | |
| None | |

#### 

#### Send a request

*Table 13: Send a request usecase specification*

|  |  |
| --- | --- |
| **Name** | Send a request |
| **Brief Description** | The way student provide their reason and information of day off. |
| **Actor(s)** | Student |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the student go to Request page   1. User access Request site 2. Click on button ‘Add request’ 3. The user fills the form. 4. Submit the form to the system. | |
| **Alternate Flows** | |
| **Title** | **Description** |
|  |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | User successfully accessed the “Request”.  User fills all field in request information form. |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | User send request successfully. |
| Failure | Data provided incorrectly, so system fails in send. |
| **Extension Points** | |
| None | |

#### Check attendance

*Table 14: Check attendance usecase specification*

|  |  |
| --- | --- |
| **Name** | Check attendance |
| **Brief Description** | The way all users check attendance |
| **Actor(s)** | All users |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the user chooses the function “Attendacne”   * + - 1. User access Attendacne site.       2. The user select the "Project name " in the system.       3. The user presses the "Punch out" button. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Check attendance successful. |
| Failure | Check attendance failed. |
| **Extension Points** | |
| None | |

#### View student/ request/ project information.

*Table 15: View student/ request/ project usecase specification*

|  |  |
| --- | --- |
| **Name** | View student/ request/ project information |
| **Brief Description** | The way teacher view student/ request/ project information. |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to view student/ request/ project information.   1. Teacher choose the item that they want to read more detail or click on the “View detail” button 2. The system redirects the user to the view the detail. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | The teacher successfully logged into the system. |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | View successfully. |
| Failure | View failed. |
| **Extension Points** | |
| None | |

#### Edit student/ request/ project information.

*Table 16: Edit student/ request/ project information usecase specification*

|  |  |
| --- | --- |
| **Name** | Edit student/ request/ project information. |
| **Brief Description** | The way teacher edit student/ request/ project information. |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to edit.   1. The teacher chooses the "edit" button in the system. 2. The system displays the “General information” form. 3. Teacher re-enters new information and clicks the " Save Changes" button. 4. The notification system has successfully changed information. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User confirmation of new information failed | 1. The system requires the user to enter new information. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | The user successfully logged into the system. |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Change successfully, user change information successfully. |
| Failure | The change failed, user change information failure. |
| **Extension Points** | |
| None | |

#### Delete student/ request/ project.

*Table 17: Delete student/ request/ project specification*

|  |  |
| --- | --- |
| **Name** | Delete student/ request/ project. |
| **Brief Description** | The way teacher delete student/ request, project. |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to delete:   1. Teacher chooses the “delete student” or “delete request” or “delete project” function in the system. 2. Teacher chooses an object to delete and clicks the "Delete" button. 3. The notification system has successfully changed information. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| User confirmation of new information failed | 1. The system requires the user to enter new information. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | The user successfully logged into the system with the type Teacher. |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Delete successfully, teacher deletes. |
| Failure | Delete failed, teacher deletes information failure. |
| **Extension Points** | |
| None | |

#### Add student/ request/ project.

*Table 18: Add student/ request/ project specification*

|  |  |
| --- | --- |
| **Name** | Add student/ request/ project to the database |
| **Brief Description** | The way teacher add student, request and project to system |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to add student, request and project to the system.   1. Teacher fill in the information of their option that they want to add to the system. 2. The system displays the relevant input field of the student, request, voucher and project that the teacher has to provide. 3. Teacher press the "Add” button. 4. The notification system has been successfully added. | |
| **Alternate Flows** | |
| **Title** | **Description** |
| Fill out the information missing | 1. The system requires re-fill and full information of the student, request and project. |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  | The teacher successfully login into the website and they are on their manage page |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Add successfully, teacher add to the system successfully. |
| Failure | Add failed, teacher add to system failure. |
| **Extension Points** | |
| None | |

#### Acceptance request

*Table 19: Acceptance request usecase specification*

|  |  |
| --- | --- |
| **Name** | Confirm request |
| **Brief Description** | The way teacher check the user request confirm this |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to manage Request.   1. Admin chooses the " Request " site in the system. 2. The system displays the Manage table. 3. Admin chooses the Approve/ Decline request button. 4. The system sends to the user a notification confirmation. | |
| **Alternate Flows** | |
| **Title** | **Description** |
|  |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Acceptance request successfully, student request approved. |
| Failure | Acceptance request failed. |
| **Extension Points** | |

1. Create task

Table 20: Create task usecase specification

|  |  |
| --- | --- |
| **Name** | Create task |
| **Brief Description** | The way teacher create task for student |
| **Actor(s)** | Teacher |
| **Flow of Events** | |
| **Basic Flow** | |
| This usecase starts when the teacher chooses the function to manage Tasks.   * + - 1. Teacher chooses the " Tasks" site in the system.       2. Teacher chooses the " Create task" button.       3. Teacher fill out all the information of form.       4. Click Create button. | |
| **Alternate Flows** | |
| **Title** | **Description** |
|  |  |
| **Pre-Conditions** | |
| **Title** | **Description** |
|  |  |
| **Post-Conditions** | |
| **Title** | **Description** |
| Success | Create task successfully. |
| Failure | Create task failed. |
| **Extension Points** | |

# **CHAPTER 4: SYSTEM DESIGN**

## **4.1. Database Design**

*Table 21: User database*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** | **Notes** |
| 1 | userNameID | Int | PRIMARY KEY | Username ID |  |
| 2 | firstname | varchar |  | First name |  |
| 3 | lastname | varchar |  | Last name |  |
| 4 | userName | varchar |  | User name |  |
| 5 | userPassword | varchar |  | User Password |  |
| 6 | dateOfBirth | date |  | Date of Birth |  |
| 7 | role | varchar |  | Type of user |  |
| 8 | userEmail | varchar |  | Email of user |  |
| 9 | userPhone | int |  | Number phone of user |  |
| 10 | address | varchar |  | Address |  |
| 11 | emailVerified | boolean |  | Email verified |  |

*Table 22: Project database*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** | **Notes** |
| 1 | userID | int | Foreign key | Username ID |  |
| 2 | projectName | varchar |  | Project Name |  |
| 3 | projectDescription | varchar |  | Project Description |  |
| 4 | startDate | date |  | Start Date |  |
| 5 | endDate | date |  | End Date |  |
| 6 | learnDate | date |  | Learn Date |  |
| 7 | attendanceAfterMinute | number |  | Attendance After Minute |  |
| 8 | createdBy | int |  | Created By |  |
| 9 | maxJoin | int |  | Max Join |  |
| 10 | joined | int |  | Joined |  |
| 11 | totalLesson | int |  | Total Lesson |  |

*Table 23: Request database*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** | **Notes** |
| 1 | requestID | int |  | Request ID |  |
| 2 | projectId | int | Foreign key | Project id |  |
| 3 | userId | int |  | User Id |  |
| 4 | date | date |  | Date |  |
| 5 | proof | varchar |  | Proof |  |
| 6 | reason | varchar |  | Reason |  |
| 7 | status | string |  | Status |  |
| 8 | type | string |  | Type |  |
| 9 | approver | string |  | Approver |  |

*Table 24: Task database*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** | **Notes** |
| 1 | taskID | int |  | Task id |  |
| 2 | projectId | int | Foreign key | Project Id |  |
| 3 | submmited | array |  | Submmited |  |
| 4 | files | Array{string} |  | Files |  |
| 5 | description | varchar |  | Description |  |
| 6 | startTime | date |  | Start Time |  |
| 7 | endTime | date |  | End Time |  |
| 8 | createdBy | int |  | Created By |  |

*Table 25: Student join database*

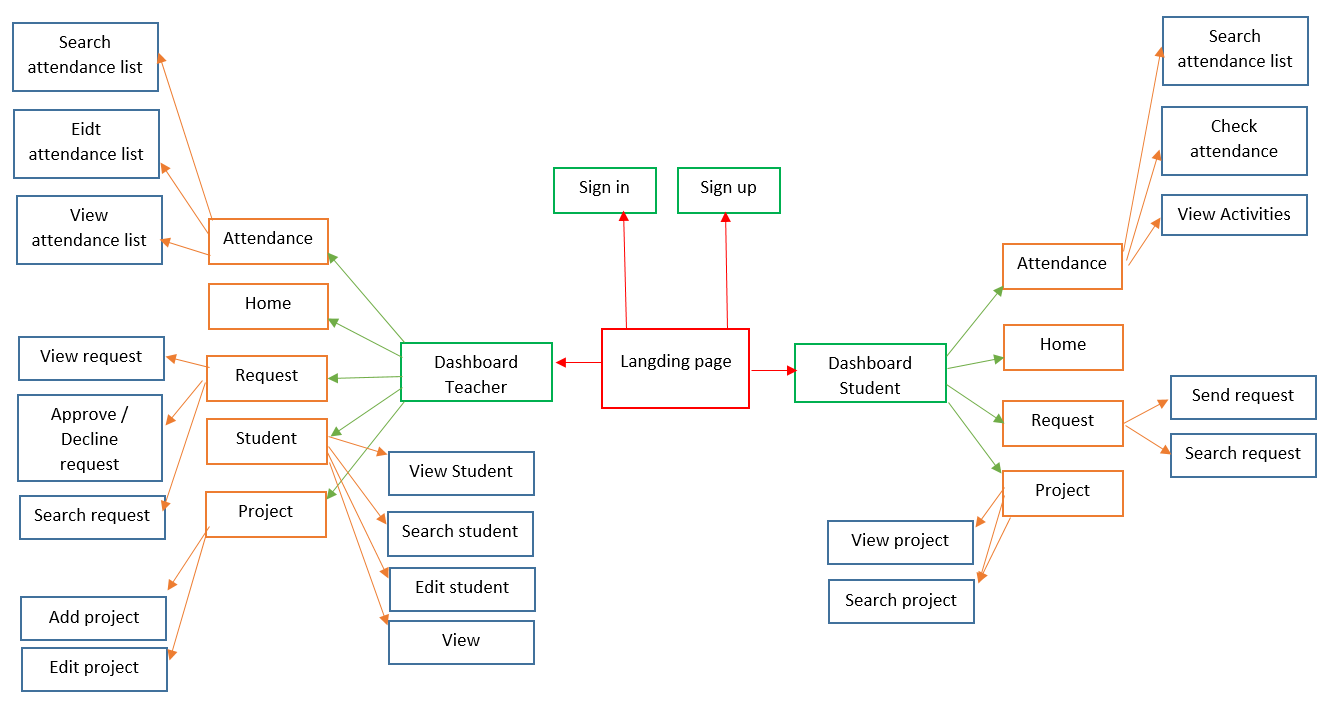
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** | **Notes** |
| 1 | projectId | int | Foreign key | Project Id |  |
| 2 | studentsJoined | array |  | Students Joined |  |
| 3 | studentID | int | Foreign key | Student id |  |

*Table 26: studentSchedules database*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Attribute** | **Type** | **Range** | **Meaning** |
| 1 | studentsJoined | int |  | Students Joined |
| 2 | schedules | array |  | Schedules |

## **4.2. UI Design**

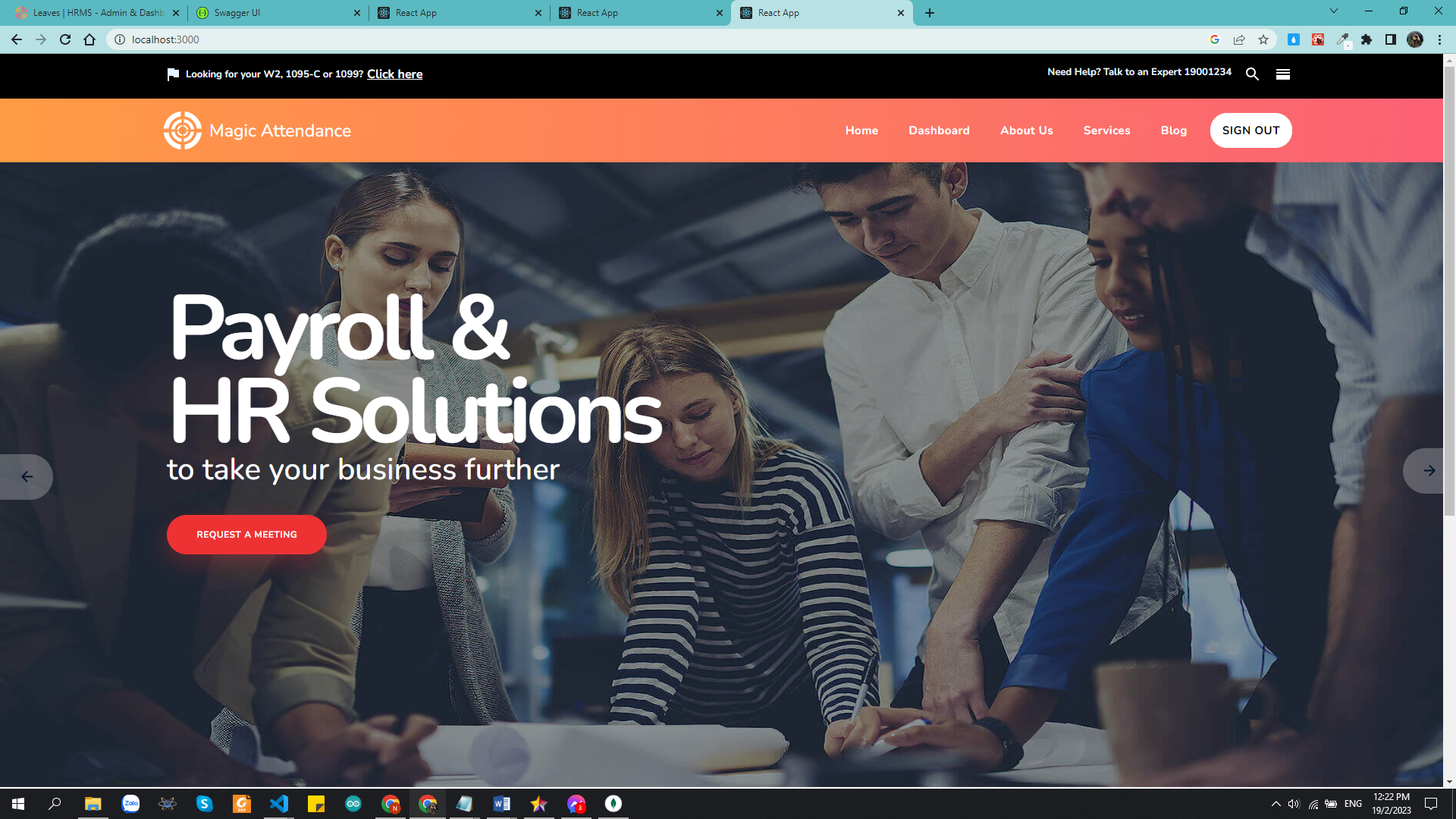
### List of screens and conversion diagrams

*Figure 2: List of screens diagram*

### Detailed description of the screens

#### Landing page

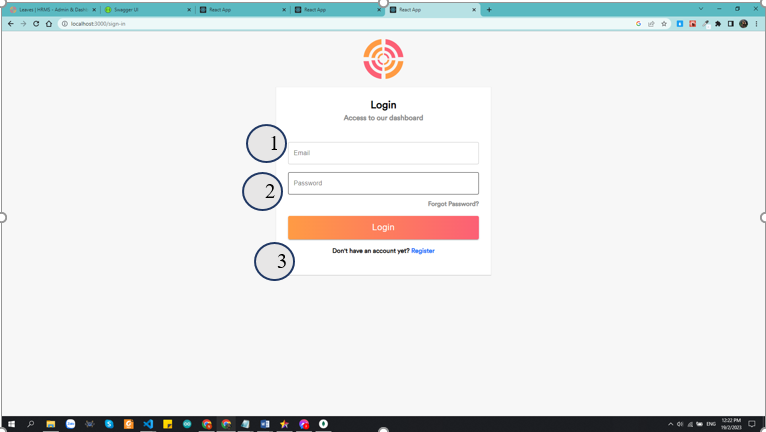
* **Meaning:** The screen designed to provide some main information of website
* **Detail:**



*Figure 3: Landing page design*

#### Sign in/ Sign up

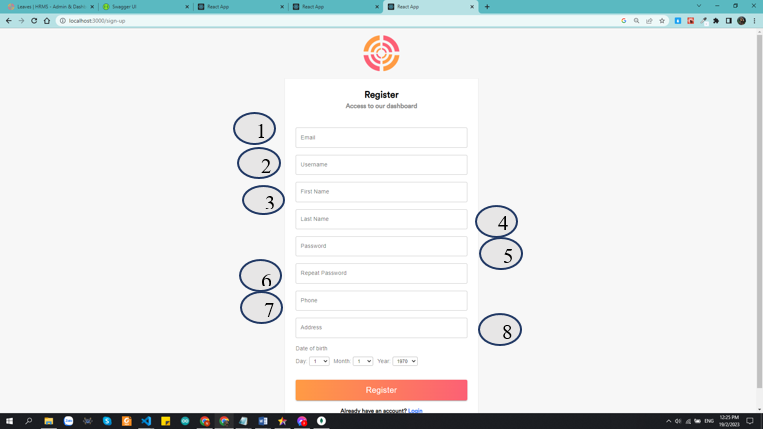
* **Meaning:** Able users to log in by their accounts or create a new one
* **Detail:**



*Figure 4: Sign in form design*

* **Objects in the screen::**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
|  | textInput | Input username or email |
|  | textInput | Input password |
|  | simpleButton | Login |



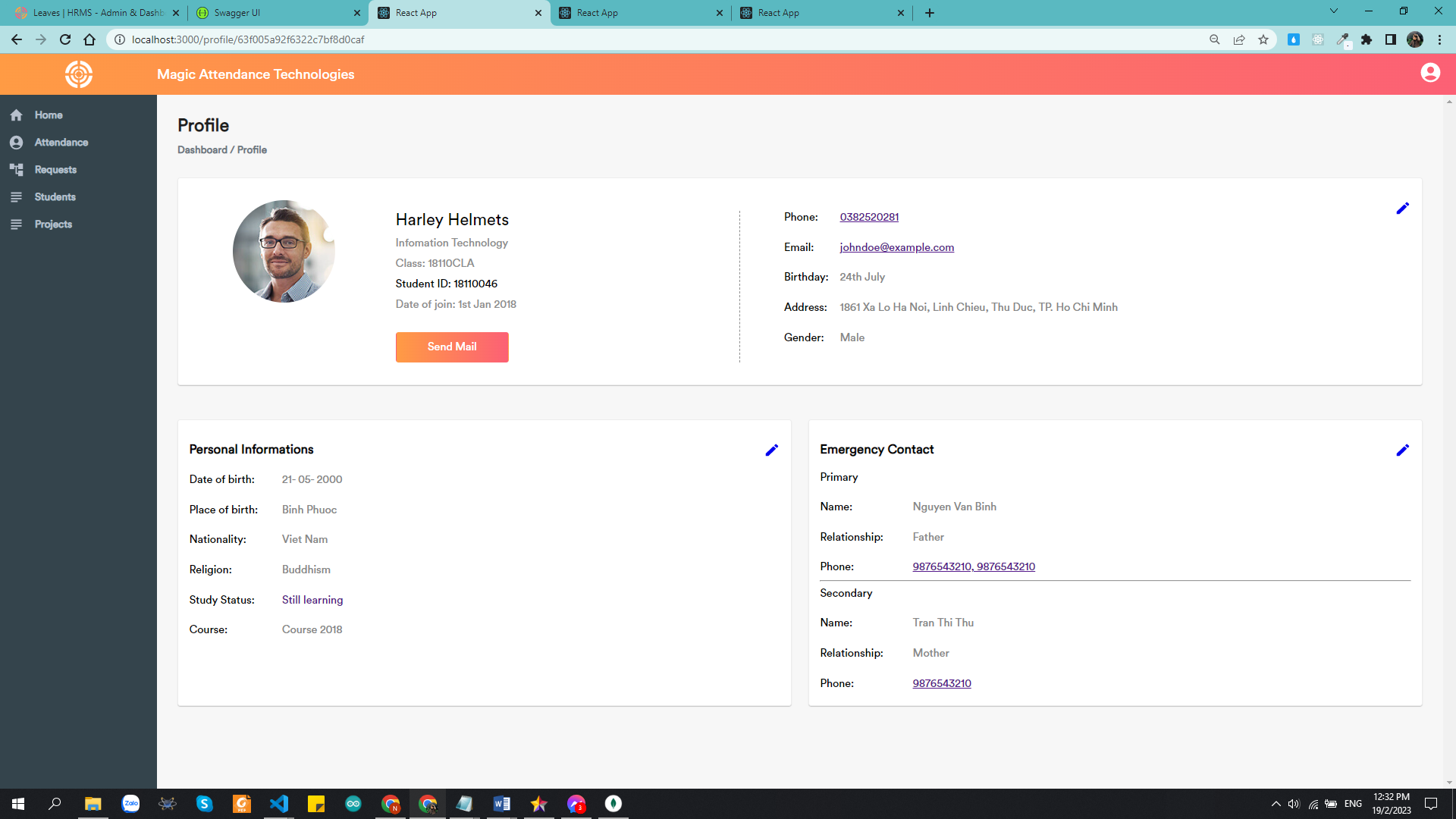
*Figure 5: Sign up form design*

* **Objects in the screen::**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
|  | textInput | Input your email |
|  | textInput | Input your username |
|  | textInput | Input your fname |
|  | textInput | Input your lname |
|  | textInput | Input password |
| 1. z | textInput | Input confirm password |
|  | textInput | Input phone |
|  | textInput | Input address |

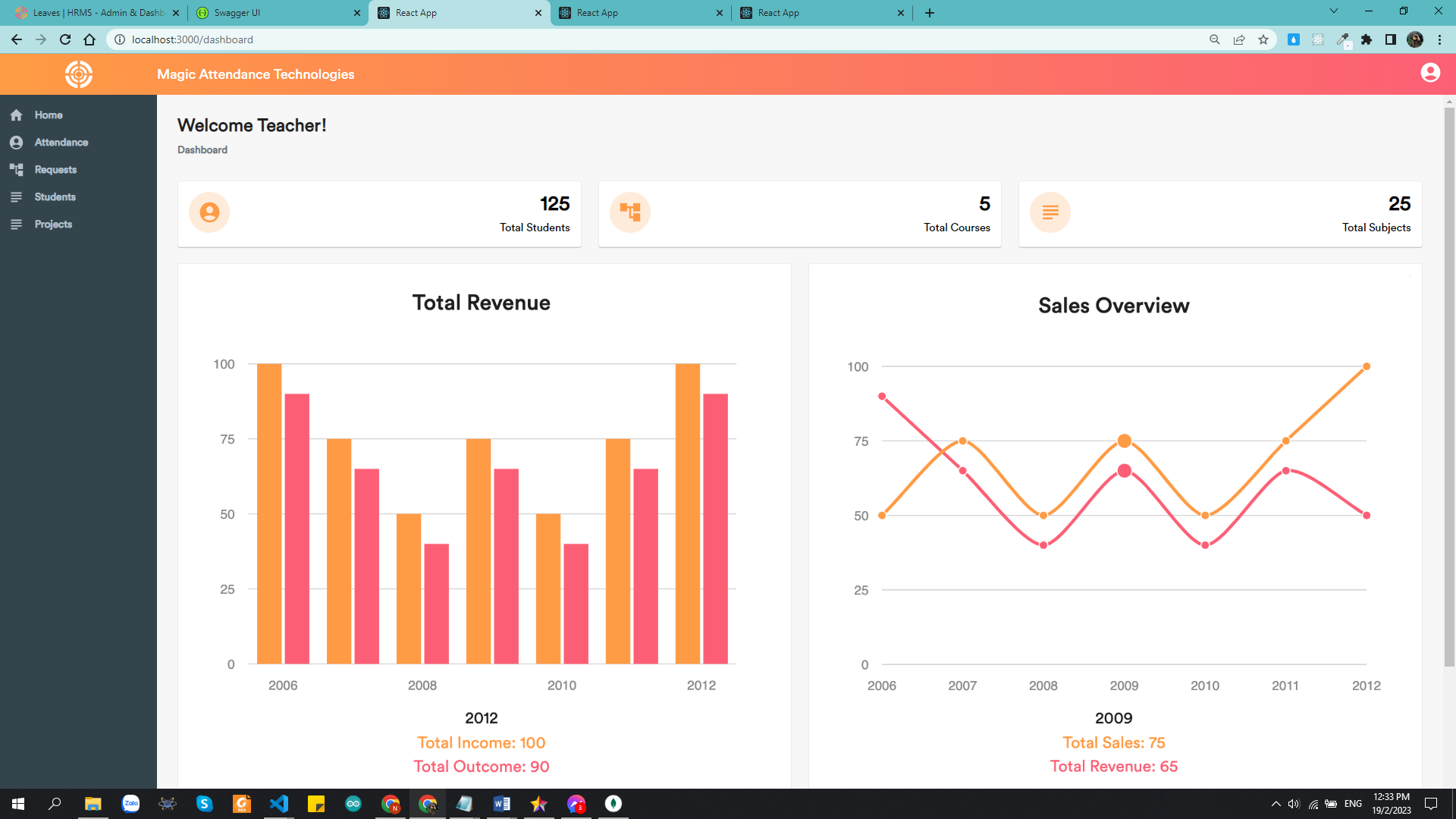
#### Personal information page

* **Meaning:** Able user can change information or password or avatar.

*Figure 6: Design personal information page*

#### Home page

* **Meaning:** Show some main information



*Figure 7: Homepage information*

#### Request page

* **Meaning:** Able user can see all features in request page

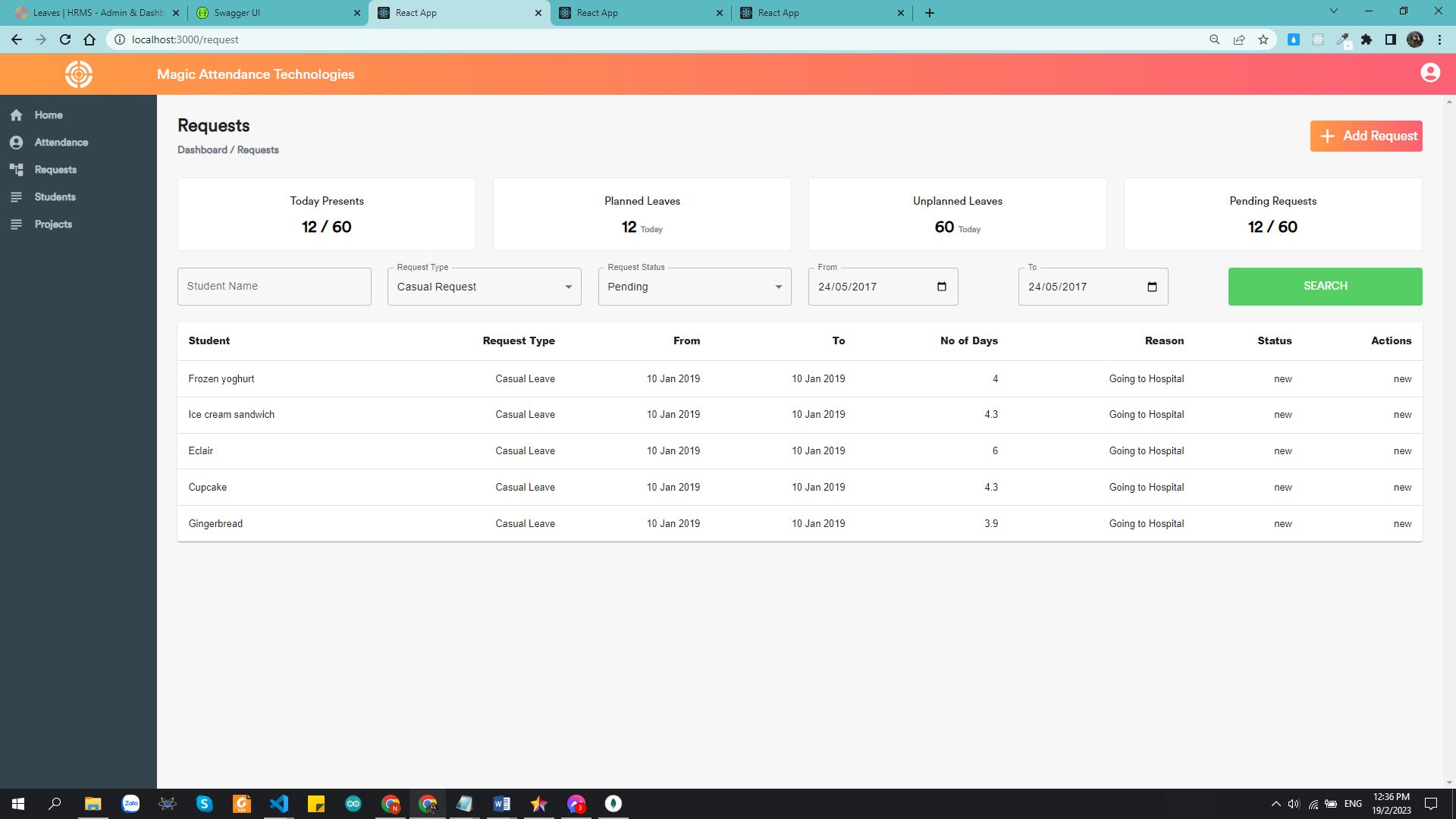


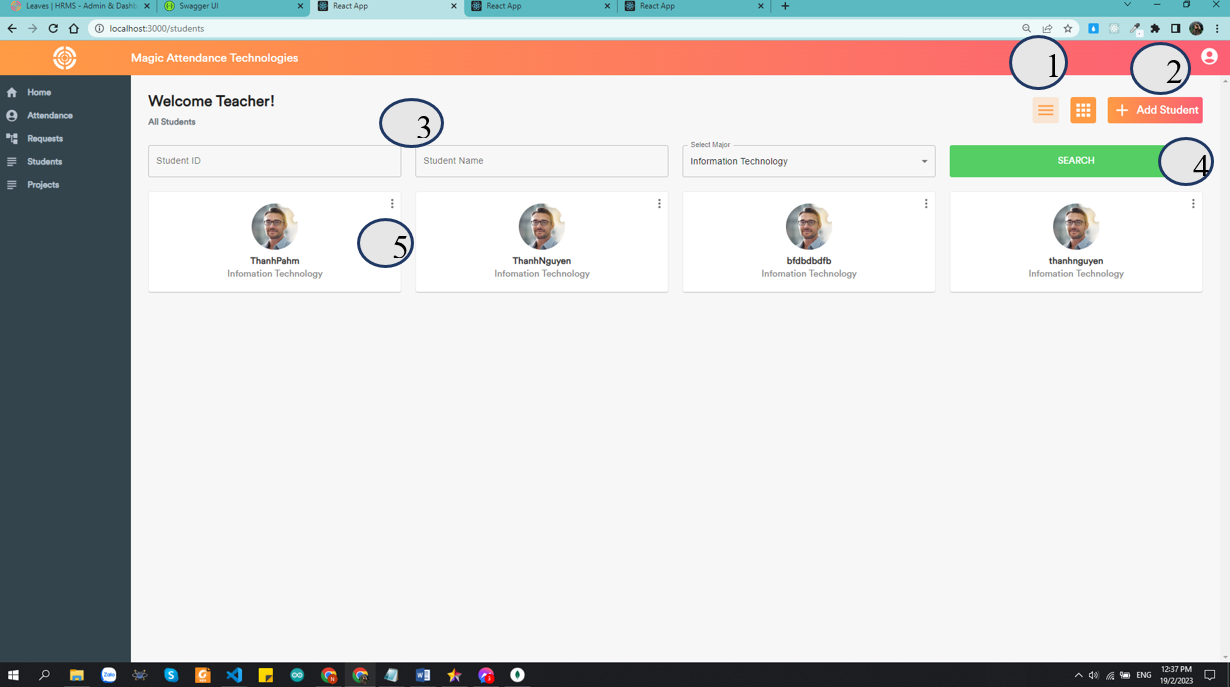
Figure 13: Request page

* **Objects in the screen::**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
| 1 | simpleButton | Add request button |
| 2 | Card | Show statistic data |
| 3 | TextField | Enter keyword to search request |
| 4 | simpleButton | Button search |
| 5 | dataTable | Show information of request |

#### All students

* **Meaning:** Able user can see all student



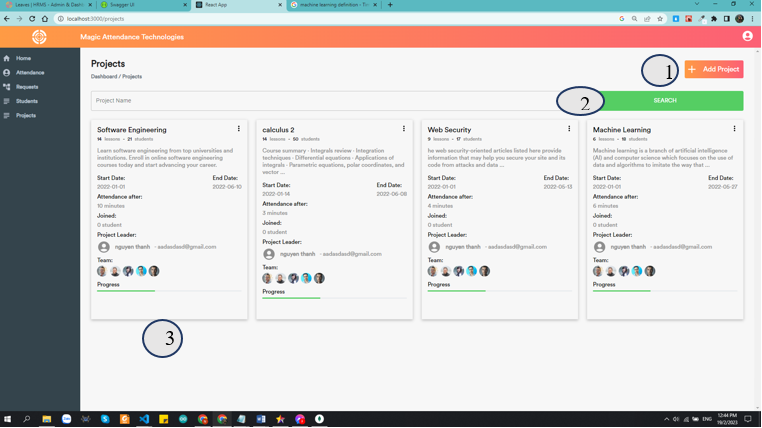
*Figure 8: All students page*

* **Objects in the screen::**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
| 1 | simpleButton | Change type of list to show |
| 2 | simpleButton | Add student button |
| 3 | TextField | Enter keyword to search student |
| 4 | simpleButton | Button search |
| 5 | simpleButton | Button show more information of student |

#### Project management page

* **Meaning:** Able user can see project page



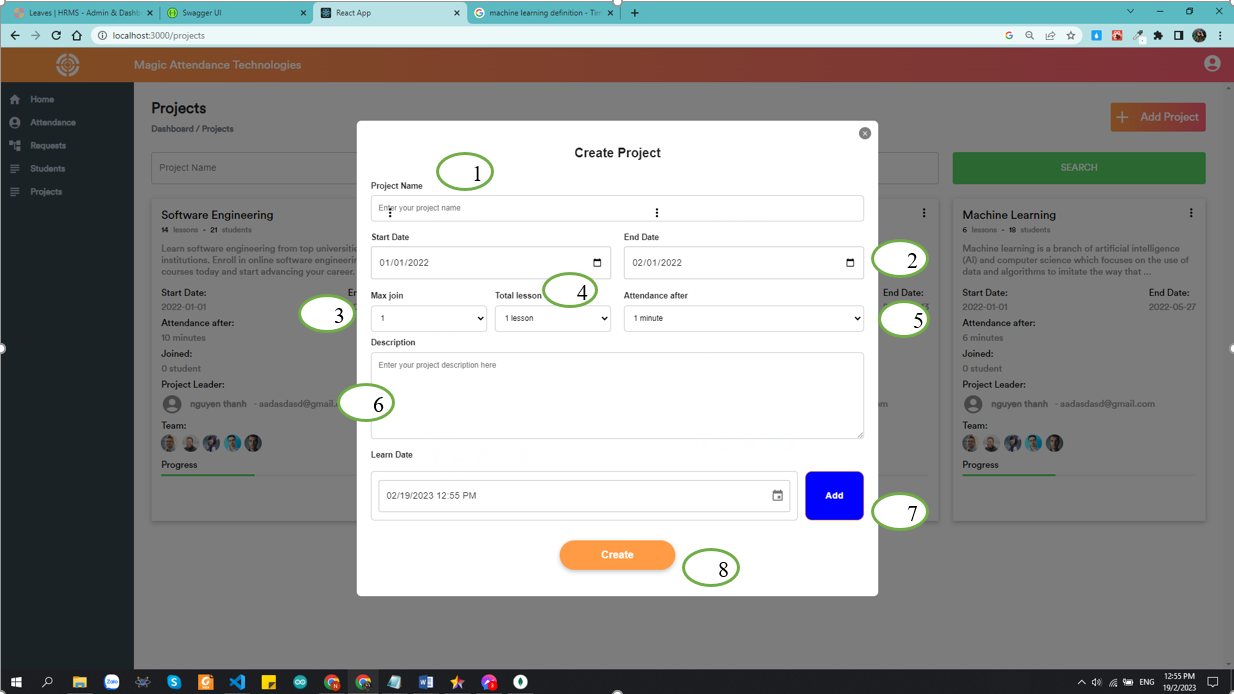
*Figure 9: Project manage page*

* **Objects in the screen::**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
| 1 | simpleButton | Add project |
| 2 | simpleButton | Search button |
| 3 | Card | Show all information about project |

#### Popup add project

* **Meaning:** Able user add project



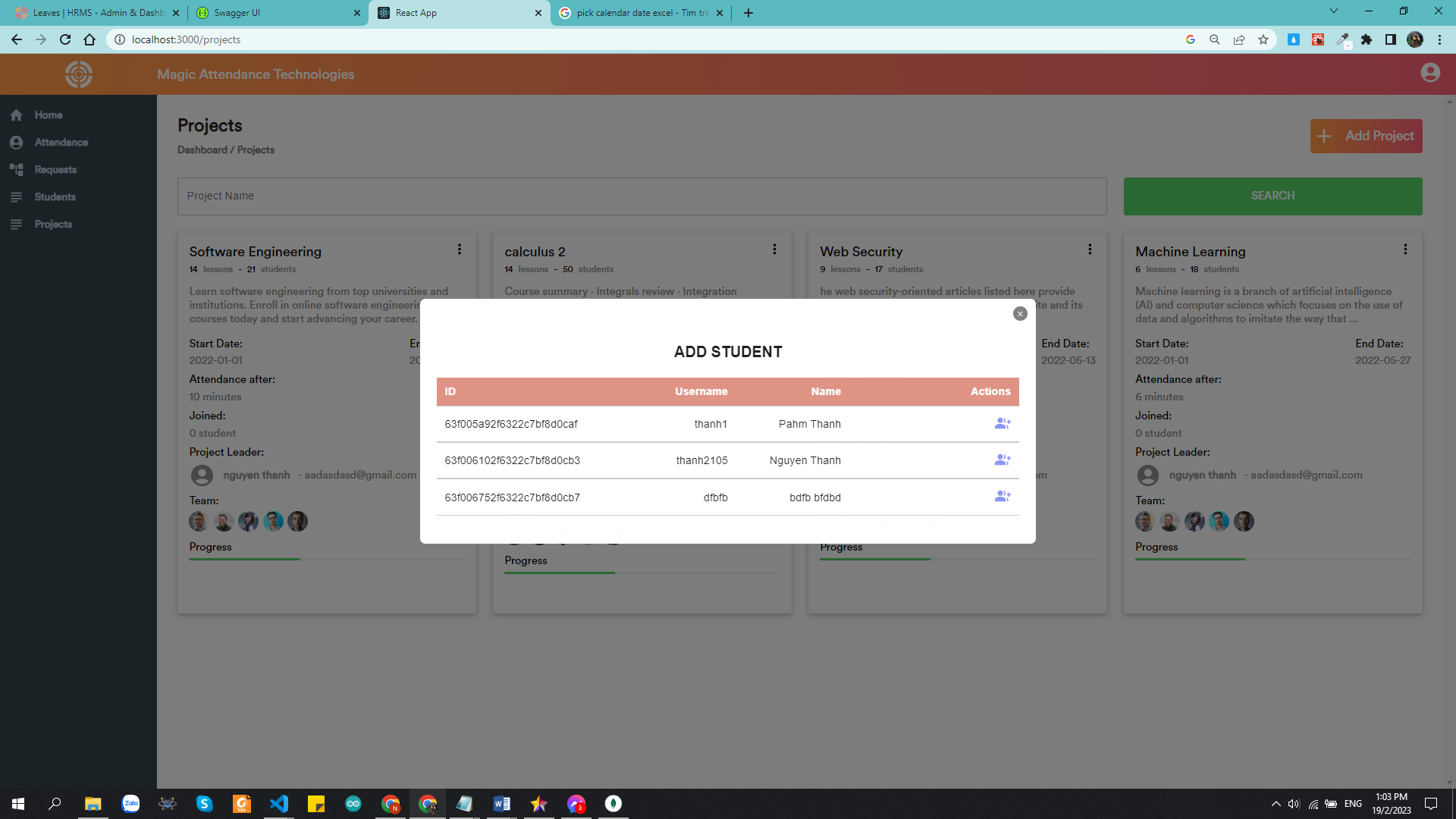
*Figure 10: Create project popup*

* **Objects on the screen:**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
| 1 | textInput | Input project name |
| 2 | dataTypeBox | Click to choose start and end day of project |
| 3 | selectBox | Choose max join in project |
| 4 | selectBox | Choose total lesson in project |
| 5 | selectBox | Choose time delay for check attendance |
| 6 | textInput | Enter decription of project |
| 7 | simpleButton | Add learn day |
| 8 | simpleButton | Create project |

#### Add student into project

* **Meaning:** Able teacher can add student into project.

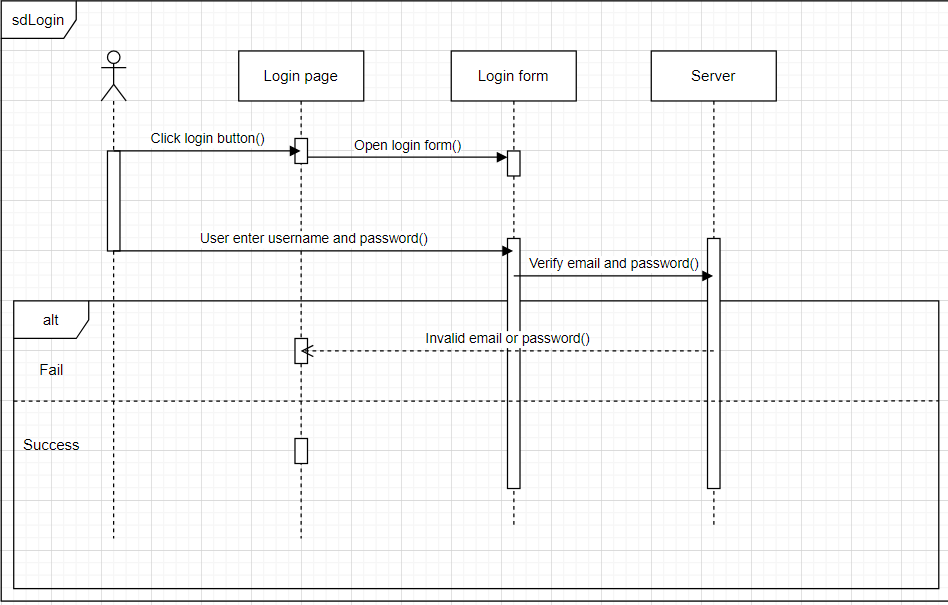


*Figure 11: Add student popup*

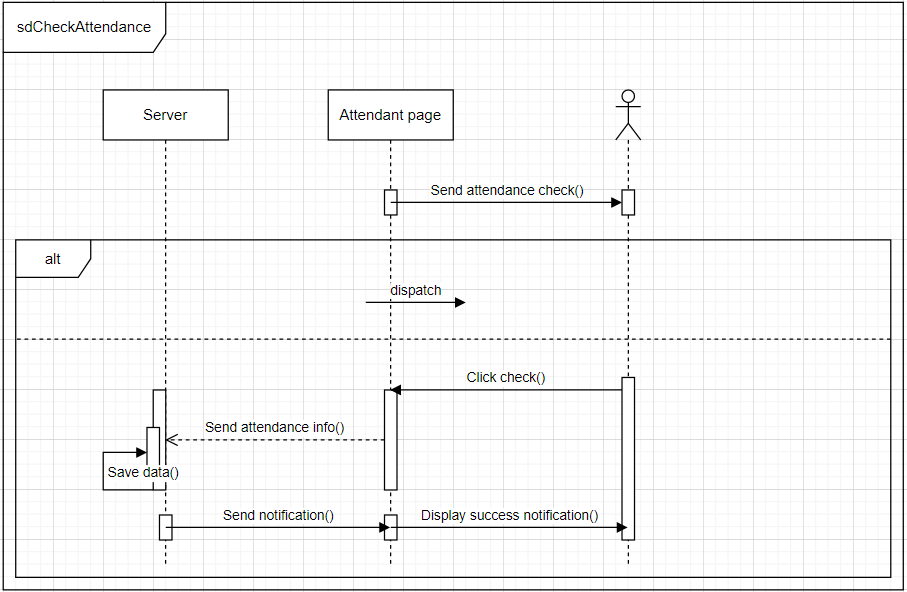
* **Objects on the screen:**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
| 1 | textInput | Show information of student |
| 2 | simpleButton | Add student to project |

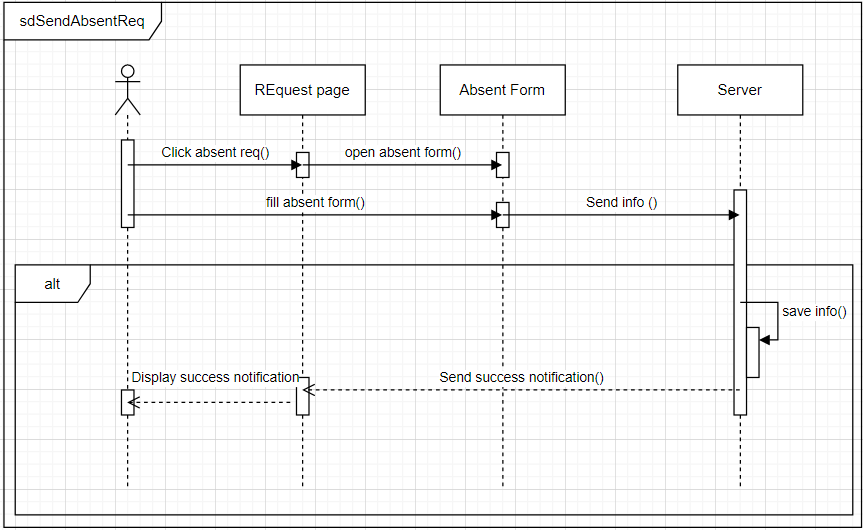
## **4.3. Sequence Diagram**



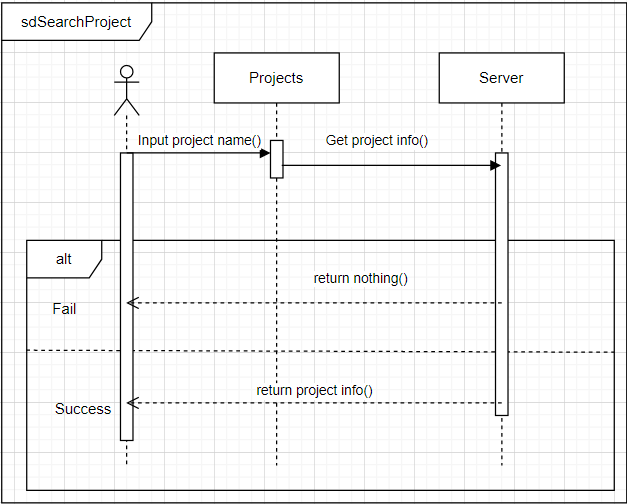
*Figure 12: Login diagram*

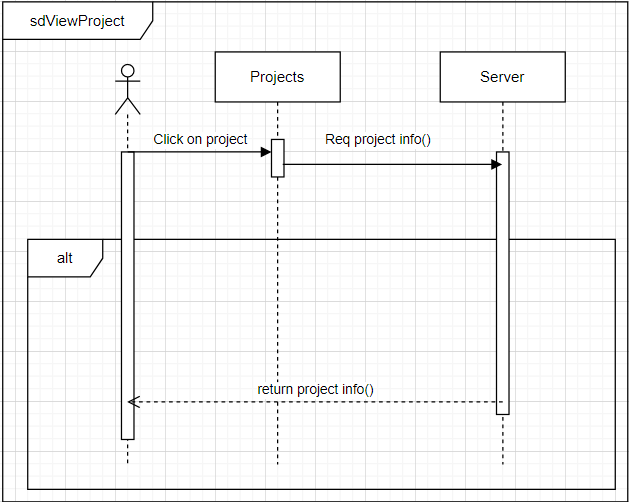


*Figure 13: Check attendance diagram*

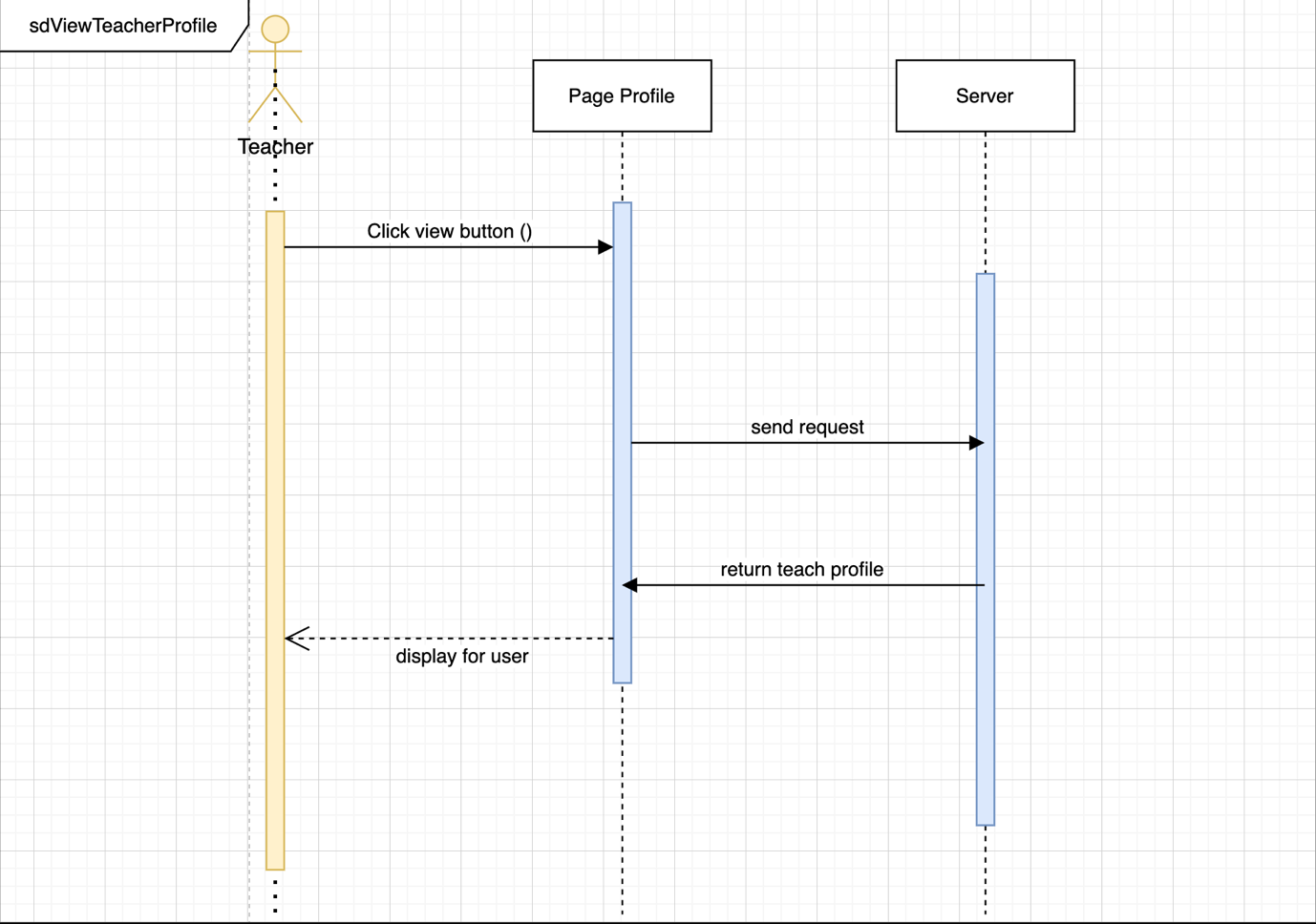


*Figure 14: Send absent request diagram*

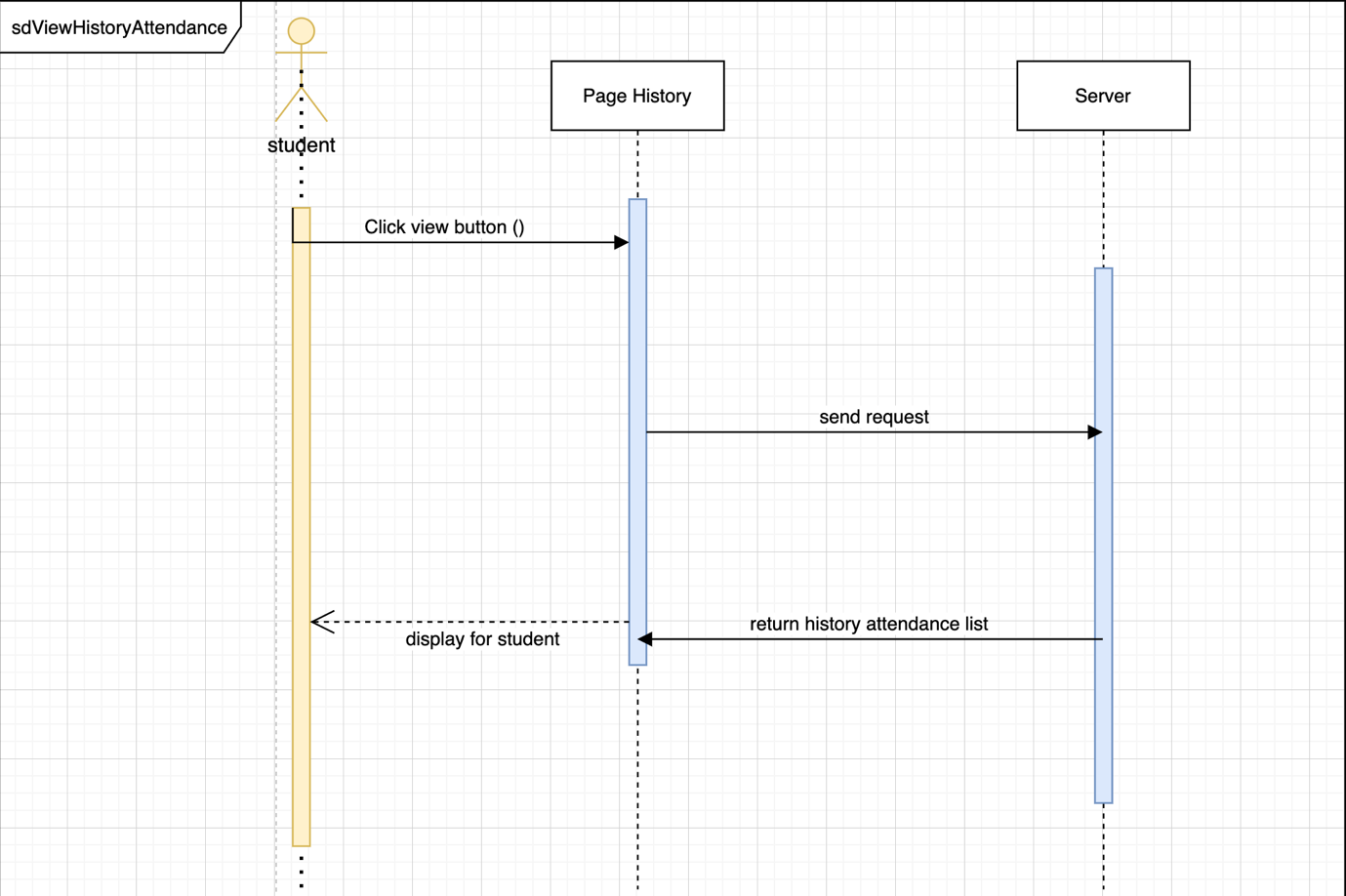
*Figure 15: Search project diagram*



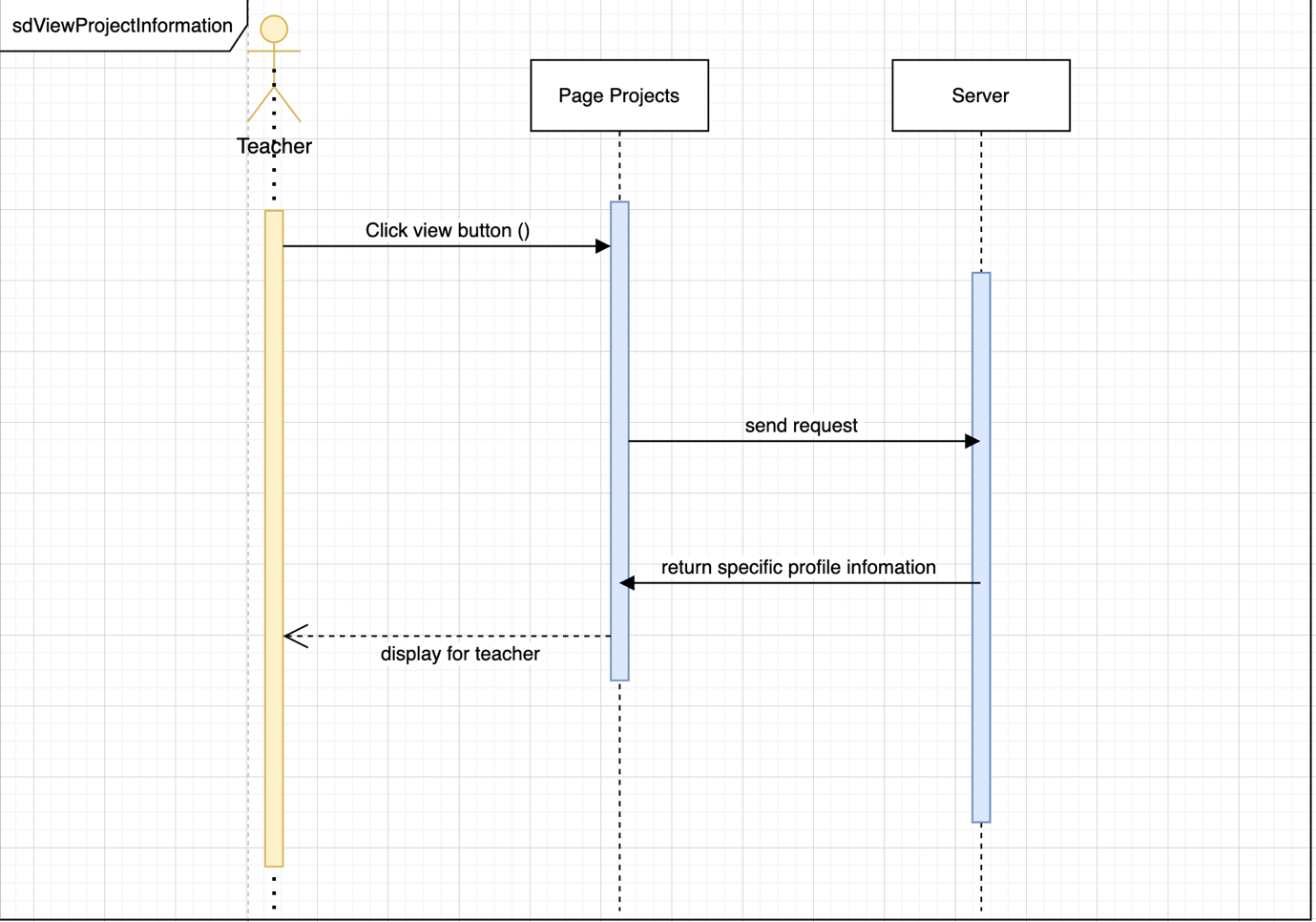
*Figure 16: View profile diagram*

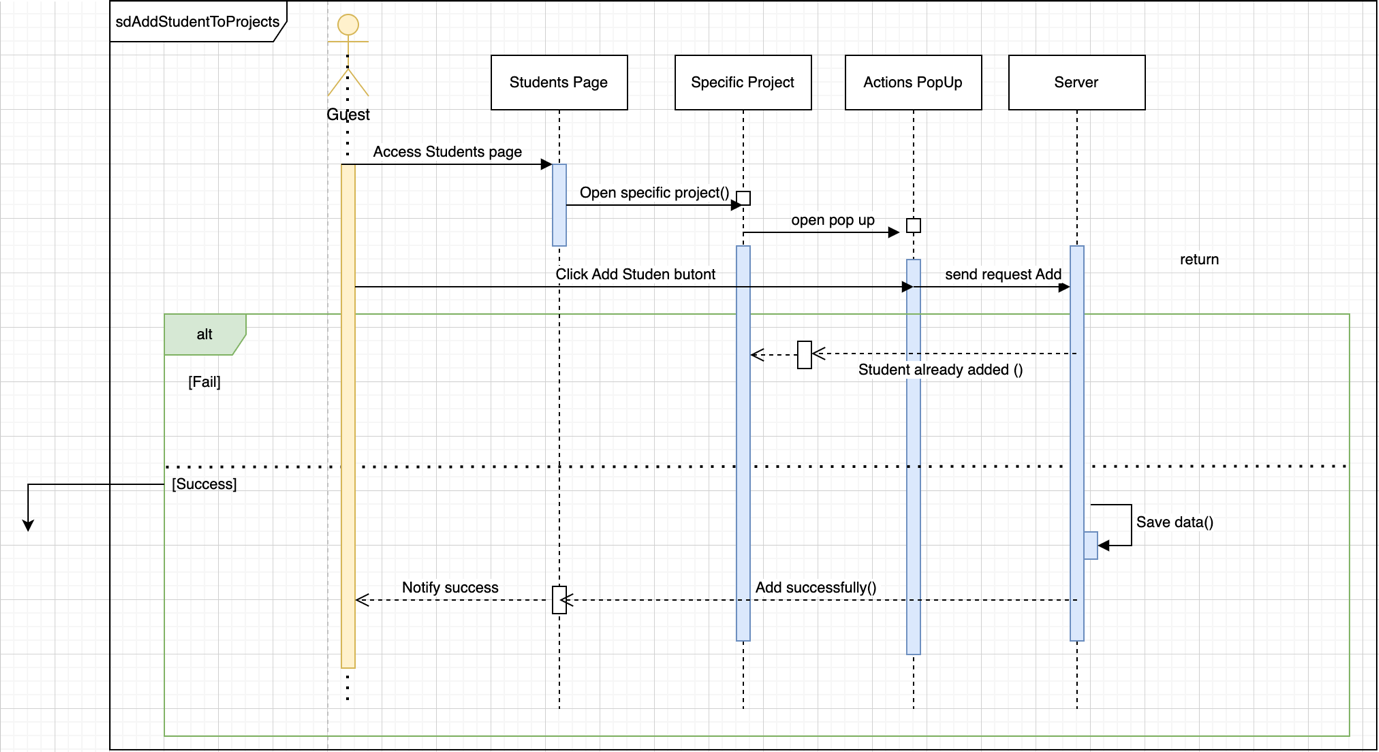


*Figure 17: View teacher profile diagram*

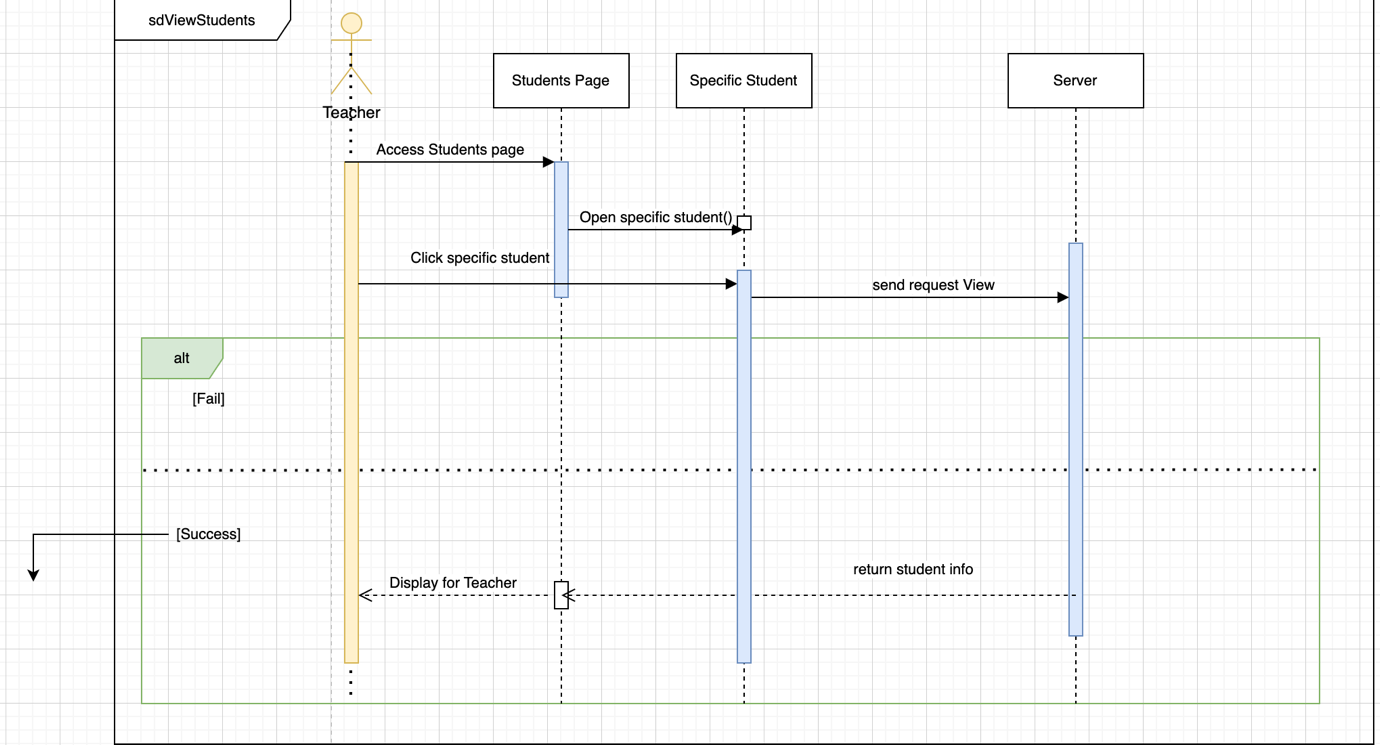


*Figure 18: View history attendance diagram*

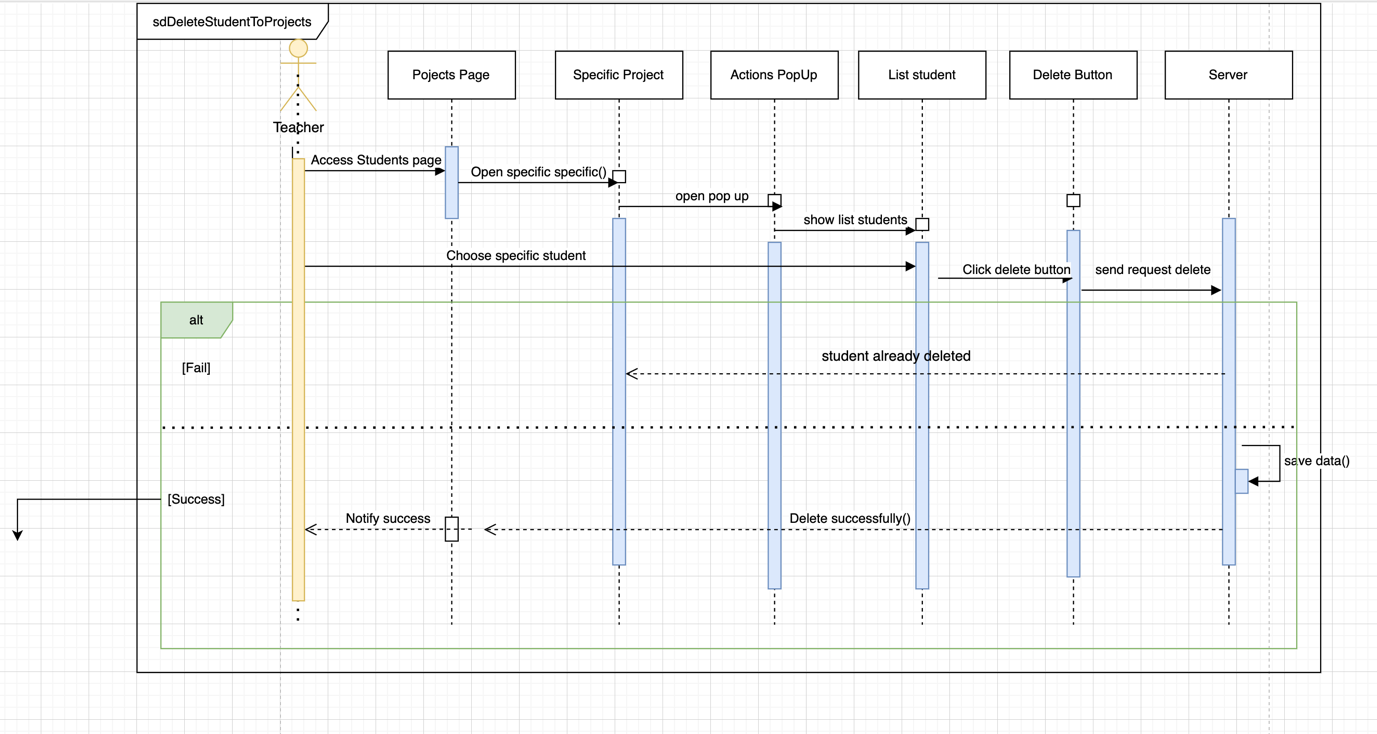
*Figure 19: View project information diagram*



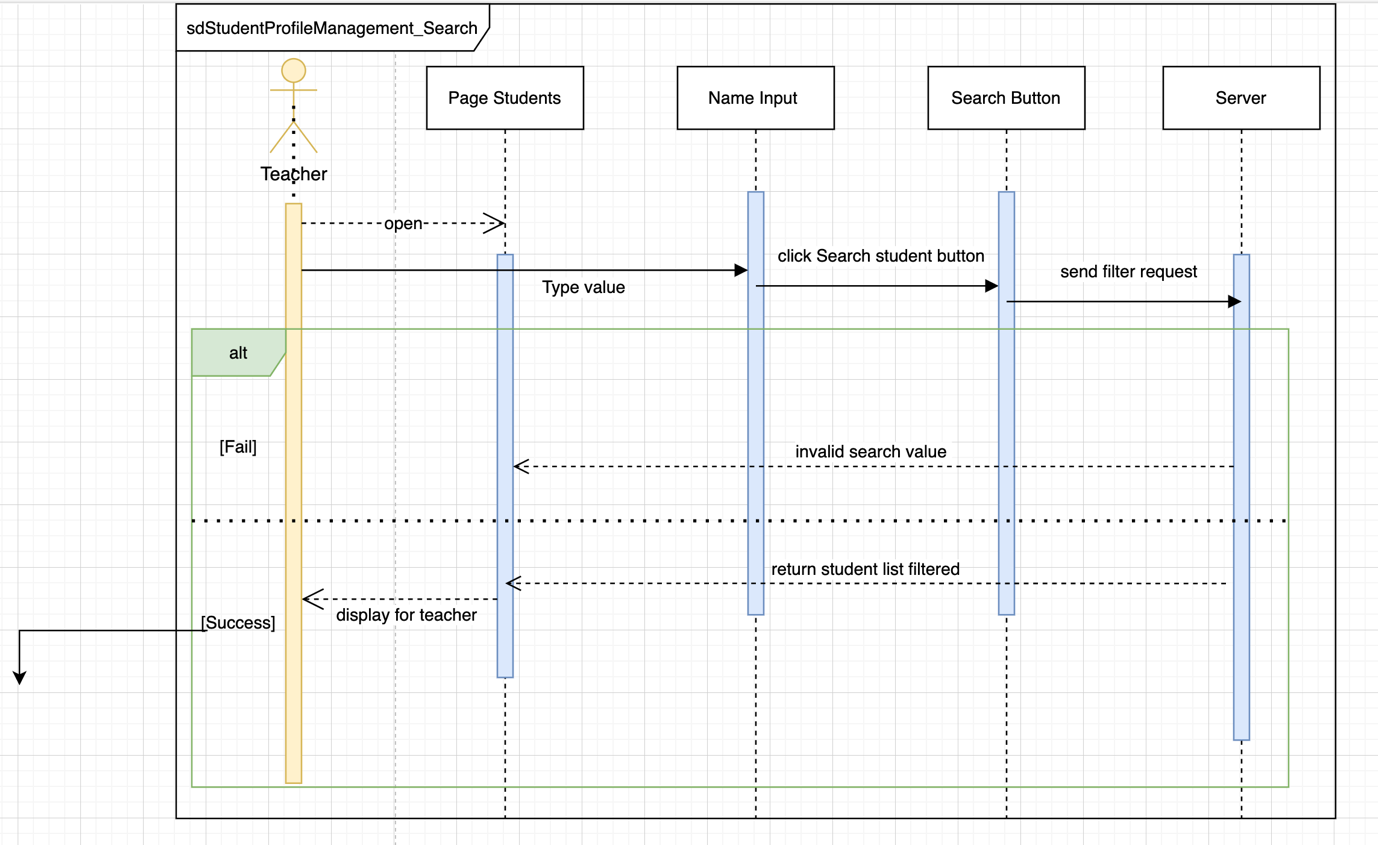
*Figure 20: Add student to project diagram*



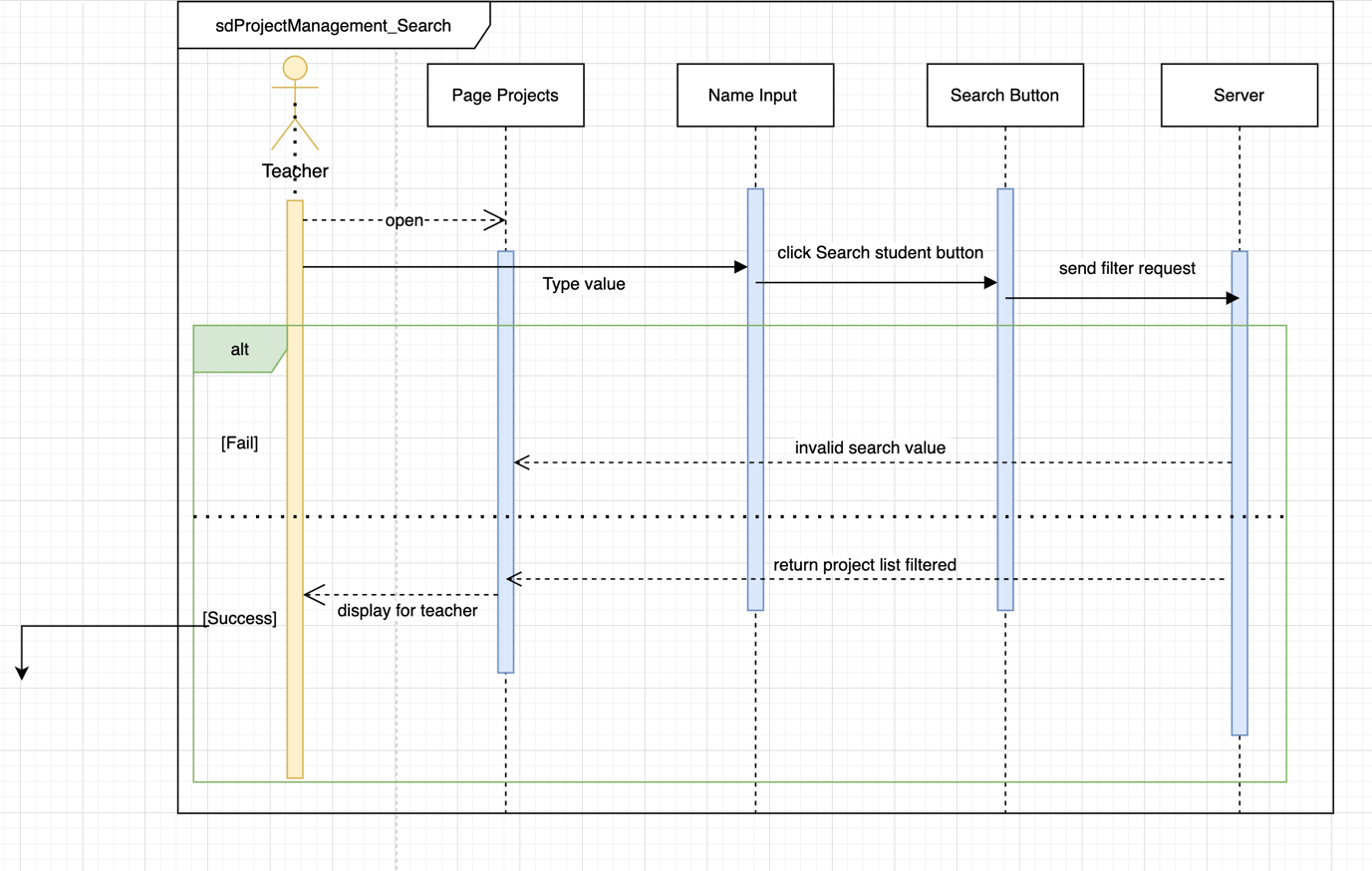
*Figure 21: View students diagram*

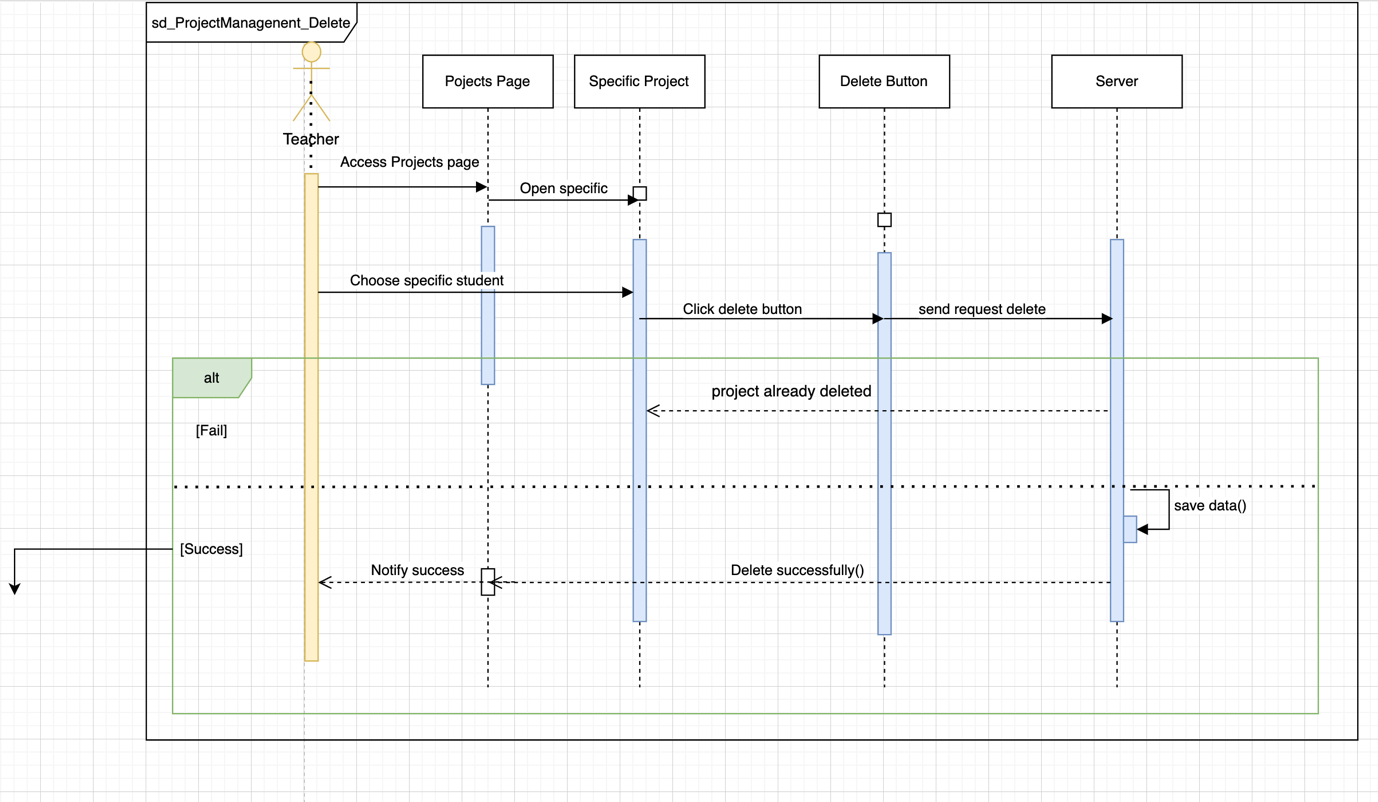


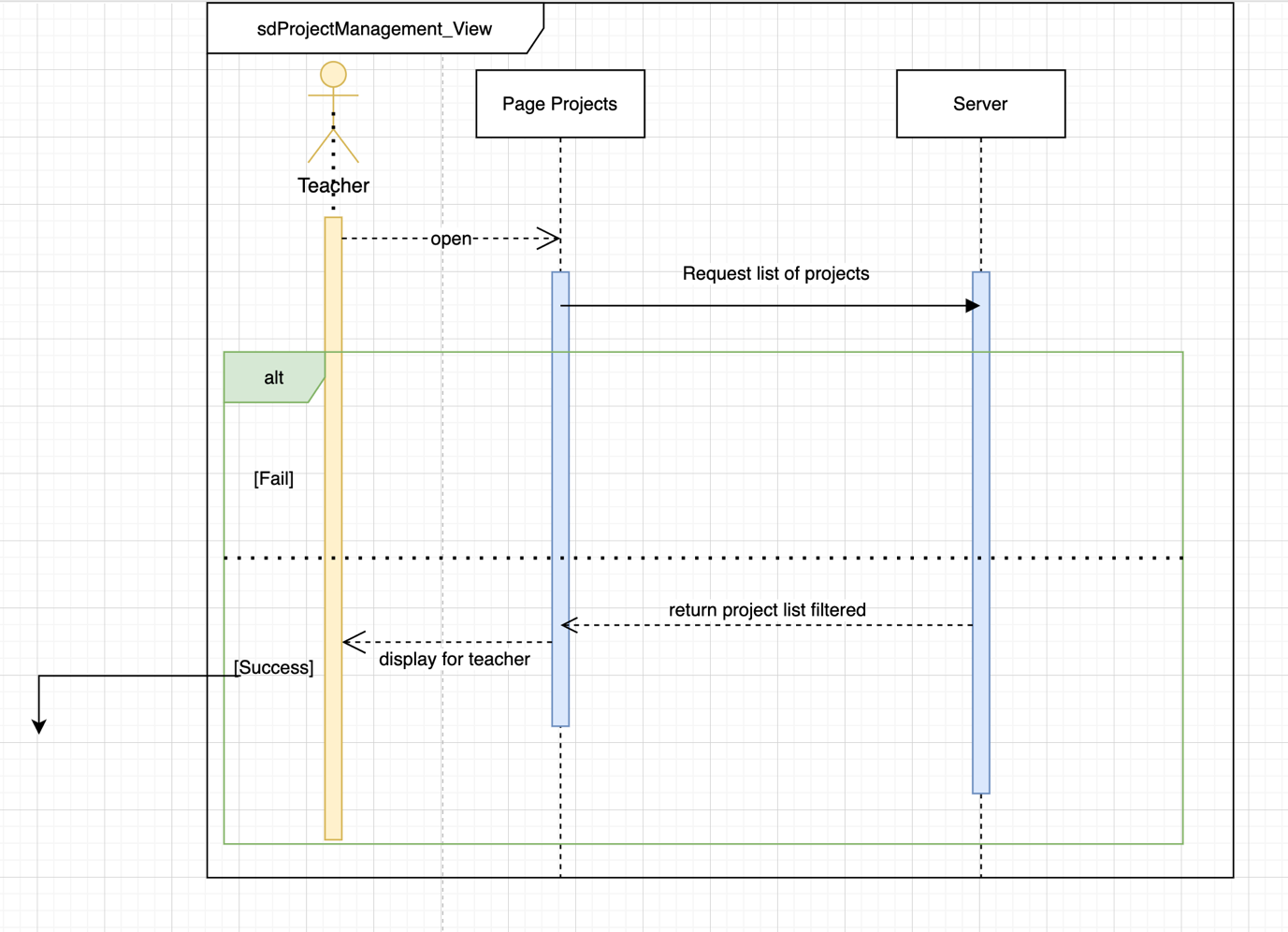
*Figure 22: Delete student diagram*



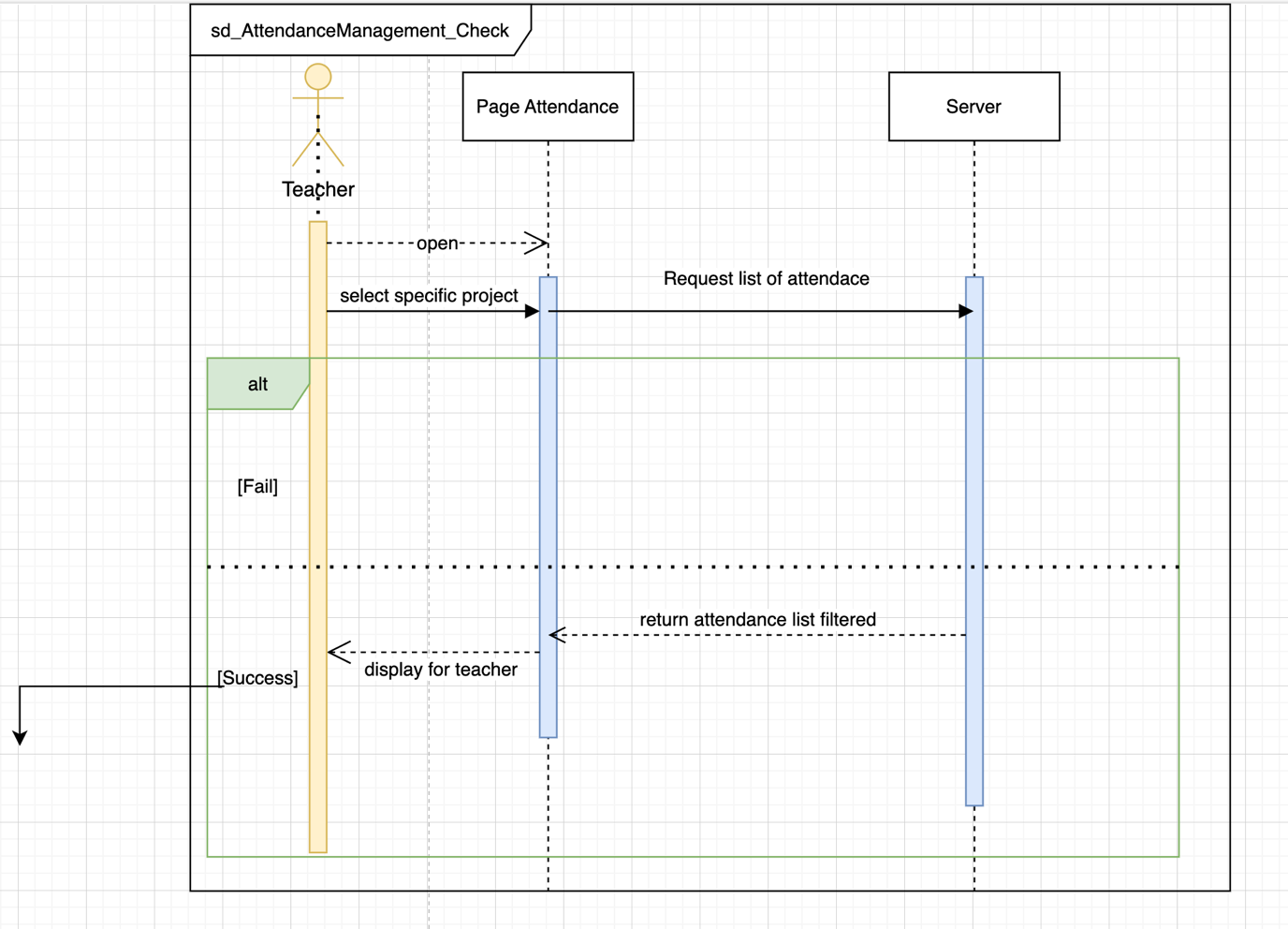
*Figure 23: Student profile diagram*

*Figure 24: Project management diagram*

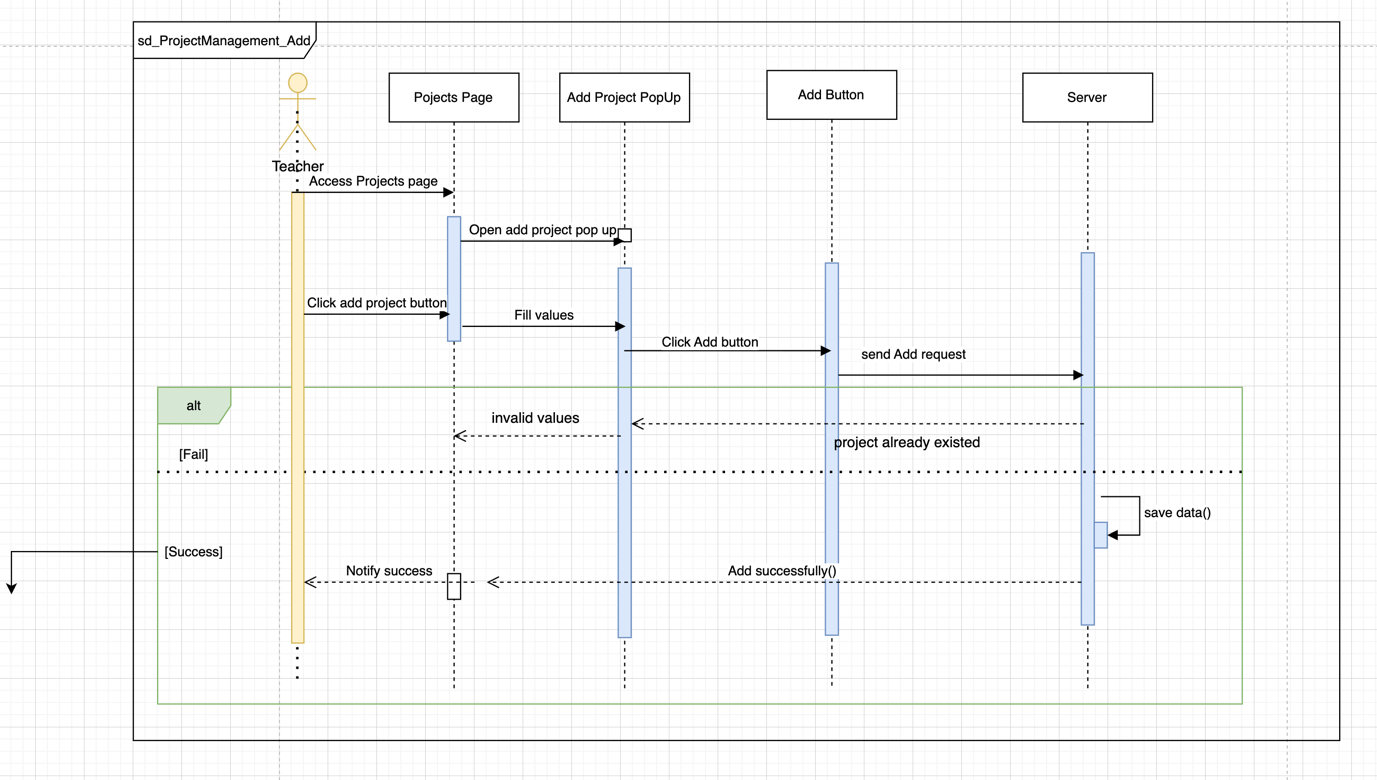
*Figure 25: Project management diagram*



*Figure 26: Project management diagram*



*Figure 27: Attendance management diagram*

*Figure 28: Project management diagram*

# **CHAPTER 5: IMPLEMENTATION AND TESTING**

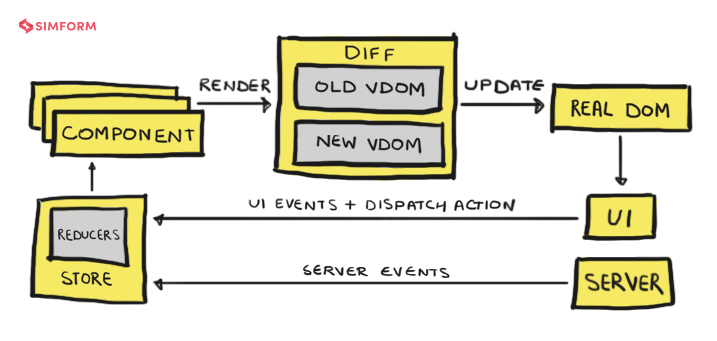
## **5.1. Implementation**

### 5.1.1. Apply the MERN stack

#### 5.1.1.1. Frontend

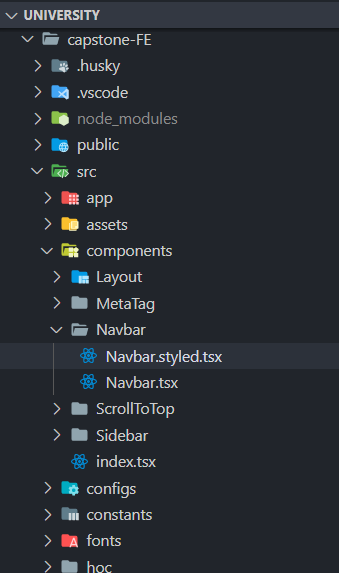
React was designed from the ground up to be gradual, and you can use as little or as much React as you want. Perhaps you just want to add some "touchpoints" to an existing page. React components are a great way to do that.

The vast majority of websites are not and need not be single-page applications. With a few lines of code and no builder, try React in a small section of your site. You can then gradually expand its presence or keep it in a few dynamic widgets.

And since it's a SPA application, we use the HTTP client for the frontend to communicate with the backend through Rest API.

*Figure 29: Front-end work*

For example, how our project applies that architectures:

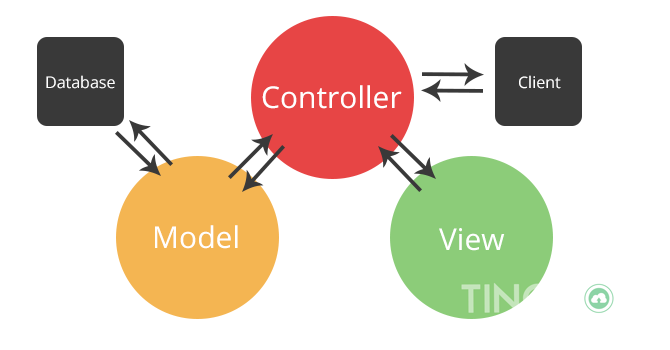


*Figure 30: Structure ReactJS in project*

#### 5.1.1.2. Backend

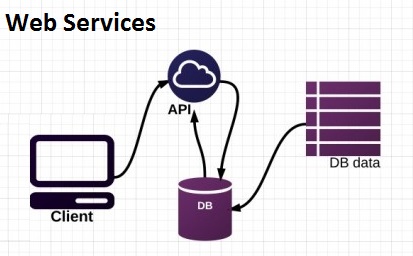
We use NodeJS for the backend and Express is the most accessible framework when using NodeJS, so learning and applying Express to the NodeJS project is a simple and effective way for our team.

Applying the MVC pattern to backend development is the simplest approach so we use the MVC pattern for the backend.



*Figure 31: MVC pattern in NodeJS*

And writing and using Rest API to communicate with the frontend is necessary for a SPA application.



*Figure 32: REST API S*

## **5.2. Testing**

### 5.2.1. Test Strategy

#### 5.2.1.1. Test Objectives

The objective of the test is to verify that the functionality of Student Attendance Website VERSION 1.0 – Student Attendance Website works according to the specifications. The test will execute and verify the test scripts, identify, fix and retest all high and medium severity defects per the entrance criteria, prioritize lower severity defects for future fixing.

#### 5.2.1.2. Test Principles

Testing will be focused on meeting the business objectives, cost efficiency, and quality. There will be common, consistent procedures for all teams supporting testing activities.

Testing processes will be well defined, yet flexible, with the ability to change as needed. Testing activities will build upon previous stages to avoid redundancy or duplication of effort.

Testing environment and data will emulate a production environment as much as possible. Testing will be a repeatable, quantifiable, and measurable activity.

There will be entrance and exit criteria.

#### 5.2.1.3. Scope and Levels of Testing

* Functional Test

**PURPOSE**: Functional testing will be performed to check the functions of the application. The functional testing is carried out by feeding the input and validating the output from the application.

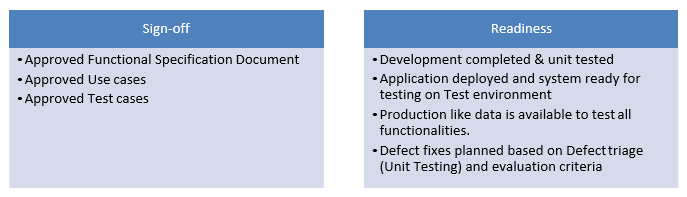
**SCOPE**: The below excel sheet details the scope of the Functional test. Note: The scope is high level due to changes in the requirement.

**TESTERS**: Our group

**METHOD**: The test case will be performed in Excel manually.

**TEST ACCEPTANCE CRITERIA**

* Approved Functional Specification document, usecase documents must be available before the start of the Test design phase.
* Test cases approved and signed-off before starting of Test execution
* Development completed, unit tested with pass status and results shared to Testing team to avoid duplicate defects
* Test environment with the application installed, configured, and ready to use state



*Table 27: Test Deliverables*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Deliverable Name** | **Author** | **Reviewer** |
| 1. | Test Plan | Our Team | Supervisor |
| 2. | Functional Test Cases | Our Team | Supervisor |
| 3. | Test Cases | Our Team | Supervisor |
| 4. | Daily/weekly status report | Our Team | Supervisor |
| 5. | Test Closure report | Our Team | Supervisor |

### 5.2.2. Test Environment

Student Attendance Website version 1.0 will be hosted at the free hosting site. Student Attendance Website version 1.0 will be hosted on three servers: One to host the backend server, one to host the frontend site, and the other to host the (database name) database.

### 5.2.3. Testing method and techniques

#### 5.2.3.1. Unit testing

* Theory: This is a type of software testing where individual units or components of the software are tested. The purpose is to validate that each unit of the software code performs as expected. A unit may be an individual function, method, procedure, module, or object.
* Classification Unit Testing. Unit Testing is of two types
* Manual
* Automated

#### 5.2.3.2. Integration Testing

* Theory: INTEGRATION TESTING is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers.
* Types of system testing strategies
* Big Bang Testing: Big Bang Testing is an Integration testing approach in which all the components or modules are integrated at once and then tested as a unit. This combined set of components is considered as an entity while testing.

Incremental Testing

* Bottom-up Integration Testing: Bottom-up Integration Testing is a strategy in which the lower-level modules are tested first. These tested modules are then further used to facilitate the testing of higher-level modules. The process continues until all modules at the top level are tested.
* Top-down Integration Testing: Top-down integration testing is a method in which integration testing takes place from top to bottom following the control flow of the software system.
* Sandwich Approach – Combination of Top-Down and Bottom-Up

Sandwich Testing is a strategy in which top-level modules are tested with lower-level modules at the same time lower modules are integrated with top modules and tested as a system..

#### 5.2.3.3. System Testing

* Theory: System testing is the testing of a complete and fully integrated product. After Integration test and Unit test.

#### 5.2.3.4. Acceptance Testing

* Theory: Acceptance Testing is a test to determine whether the software system meets the specifications or not. .
* Black Box Testing

Black Box Testing is a part of user acceptance testing. This testing method helps to analyze the functionality without allowing the tester to see the internal code structure.

* Conclusion: Acceptance Testing is one of the very important testing stages that help you evaluate the functionality of the software application to meet the specified requirements or not. Therefore, understanding what Acceptance Testing is and the types of Acceptance Testing will help you do a better job of software testing.

### 5.2.4. Black box testing and bug report

#### 5.2.4.1. Test cases

The link for access purposes:

(Note: copy and paste this link to web browser)

<https://docs.google.com/spreadsheets/d/1TRljr3YAAmz3MGQbexYW8_XEW79onNlL/edit?usp=share_link&ouid=102240459920930996528&rtpof=true&sd=true>

#### 5.2.4.2. Bug Report

|  |  |
| --- | --- |
| ID number | TC\_02 |
| Name | Space character check in 'Password' field in the Register page |
| Reporter | Hoang Vu |
| Submit Date | Jan 03, 2023 |
| Summary | The users create their information (password in this case) with an invalid value ‘123 456 78’ in any field. |
| URL | https://localhost:3000/login |
| Screenshot |  |
| Platform | Nodejs, Reactjs |
| Operating System | Windows 10 |
| Browser | Google Chrome |
| Severity | Major |
| Assigned to | Hoang Vu |
| Priority | High |

**Description**

Users try to create their account with an invalid value ‘123 89 1 ‘ but the system still allows them to accomplish their progress. There should be a form validation, also a message informs to users.

**Steps to reproduce**

> Access to website

> Move to register page

> Register with password: 123 456 78

> Click register button

**Expected result**

Show an error message

**Actual result**

Allow users to create with invalid value.

|  |  |
| --- | --- |
| ID number | FN\_04 |
| Name | Test the value in Password field with the different six special symbol |
| Reporter | Bao Tran |
| Submit Date | Jan 03, 2023 |
| Summary | The users create their information (password in this case) with an invalid value ‘;;;;;;’ in any field. |
| URL | https://localhost:3000 |
| Screenshot |  |
| Platform | Nodejs, Reactjs |
| Operating System | Windows 10 |
| Browser | Google Chrome |
| Severity | Major |
| Assigned to | Bao Tran |
| Priority | High |

**Description**

Users try to create their account with an invalid value ‘;;;;;; ‘ but the system still allows them to accomplish their progress. There should be a form validation, also a message informs to users.

**Steps to reproduce**

> Access to website

> Move to register page

> Register with password: ;;;;;;

> Click register button

**Expected result**

System show error message.

**Actual result**

There is no message, accept user to create

|  |  |
| --- | --- |
| ID number | TC\_13 |
| Name | Check for Brute Force Attacks. |
| Reporter | Tan Thanh |
| Submit Date | Jan 03, 2023 |
| URL | https://localhost:3000/login |
| Screenshot |  |
| Platform | Nodejs, Reactjs |
| Operating System | Windows 10 |
| Browser | Google Chrome |
| Severity | Medium |
| Assigned to | Tan Thanh |
| Priority | Medium |

**Description**

When the user can try many time with other username password without limited.

**Steps to reproduce**

> Access to website

> Move to login page

> Login with random username and password

> Click login button

> Try as many time

**Expected result**

Limit the number of times you enter the wrong value.

**Actual result**

Allow to try as many time as you want.

# **CHAPTER 6: CONCLUSION**

## **Achievements**

In the short time since starting to receive the topic, when going through the analysis and finding solutions to implement the topic. In general, the group has completed the goals as well as the initial expectations. Through self-study and looking for information online to do the project, my team knew and understood more about the process of building an application about ReactJS and NodeJS, database analysis, problem-solving, and teamwork. Thereby helping the group improve their skills and gain more experience.

### Theoretically achievements

My group with knowledge about:

* + - How it works and use ReactJS, NodeJS.
    - How to write and use the API, MongoDB..

### Realistic results

* + Students easily take attendance.
  + Teachers easily manage attendance.
  + Accurate attendance with location detection.
  + Remind students by email if the schedule is coming.
  + Confirm registration by email.
  + Give out homework and submit homework online.
  + The source code is available for future work.

## **Limit**

* Limited time for studying new technology.
* We don’t have much experience in testing.

## **Directions of future development**

* Improved more facial recognition features, location detection to make online attendance more accurate.
* Mark homework and summary point like the website <https://fhqx.hcmute.edu.vn> .
* Developing applications on other platforms not only on the website such as mobile and continuously improve UI and system maintenance.

# **REFERENCES**

[1**] M Satheesh, BJ D'mello, J Krol**, *Web Development with MongoDB and NodeJS*, 2015.

[2] [**Jakob Ricardo Clausen**](https://medium0.com/@jakobclausen?source=post_page-----913d307eb69--------------------------------)*, Using useMemo and useCallback in React*, April 18th 2021.

[3] **Ron Fybish***, Implementing Microservices in NodeJS*, May 20th, 2021.

[4**] Frank Joseph**, *Building microservices with Node.js*, February 1st, 2022.

[5] **Clara Ekekenta**, *Build React reusable components faster with Aspect*, March 13th, 2022.

[6] **Blake Yeboah**, *Developing MERN Stack Applications*, January 23rd, 2022.

[7] **Puneet Singh**, *OAuth implementation with Node.js and Github*, March 22nd, 2020.

[8] **Ashish Sharma**, *NodeJS and MongoDB application authentication by JWT*, March 20th, 2020.

[9] **Hridayesh Sharma**, *Password hashing in NodeJS using Bcrypt*, November 11th, 2020.

[10**] Andy Yeung**, *Using PGP Encryption with NodeJS*, November 10th, 2020.

[11] The Good and the Bad of ReactJS. Retrieved from AltexSoft website: <https://reactjs.org/tutorial/tutorial.html>

[12] Redux - Overview - Tutorialspoint. (n.d.). Retrieved from website: <https://redux.js.org/tutorials/essentials/part-1-overview-concepts>

[13] Store data with Mongodb. Retrieved from website: <https://www.mongodb.com/docs/>