

Introduction to Software Engineering

Software Design

*The student team is required to complete the **Design Document** for the assigned course project, following the attached template.*



Software Engineering Department
Faculty of Information and Technology
University of Science

Table of Contents

1	Member Contribution Assessment	2
2	Conceptual Model	3
3	Architectural Design	4
3.1	Architecture Diagram	4
3.2	Class Diagram	4
3.3	Class Specifications	5
3.3.1	Class C1	5
4	Data Design	6
4.1	Data Diagram	6
4.2	Data Specification	6
5	User Interface and User Experience Design	7
5.1	Screen Diagram	7
5.2	Screen Specifications	7
5.2.1	Screen "A"	7
5.2.2	Screen "B"	7






Software Design

Objectives

This document focus on the following topics:

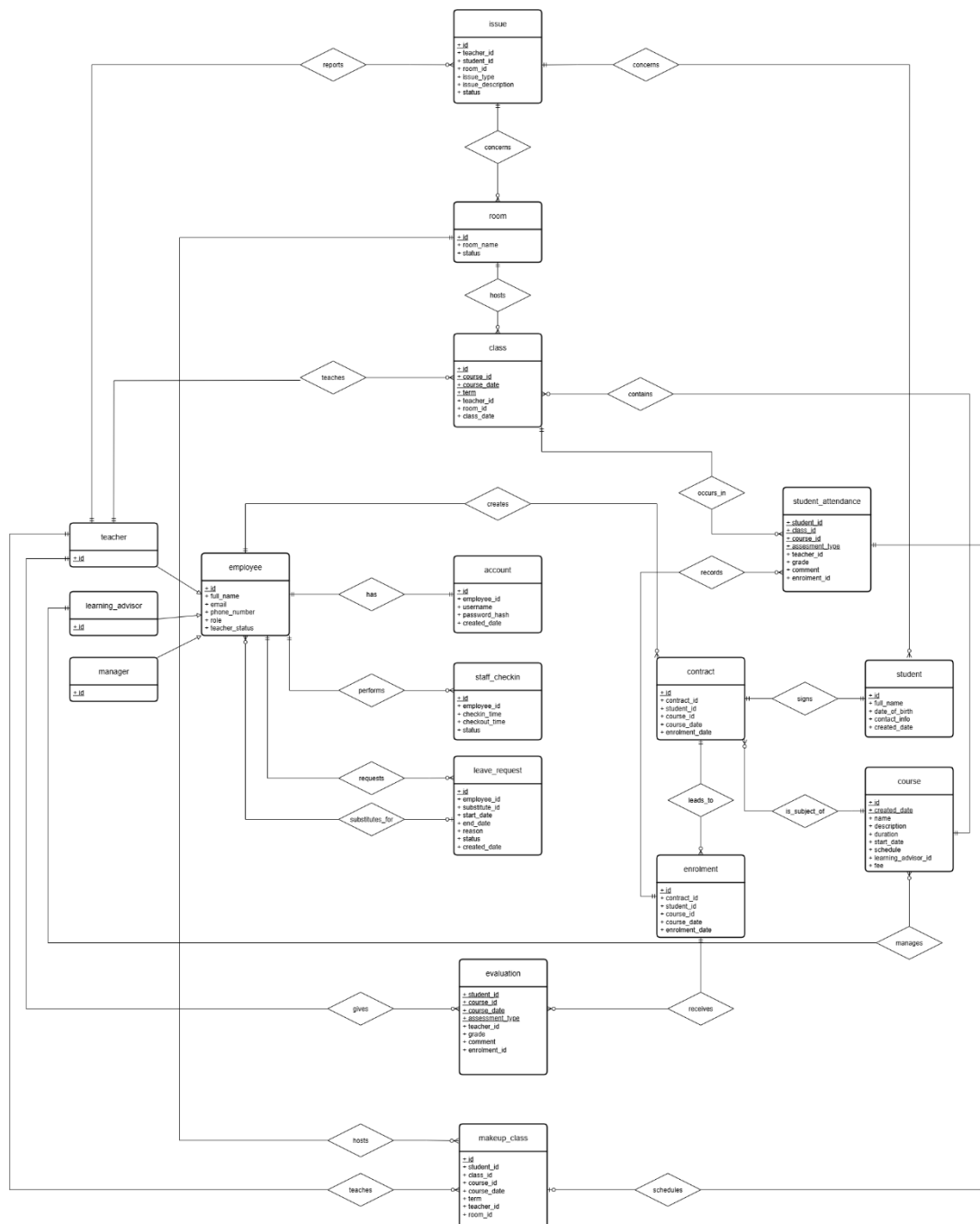
- ✓ Complete the Software Design Document with the following contents:
 - Conceptual Model
 - Architectural Design
 - Data Design
 - User Interface Design
- ✓ Understanding the Software Design Document.

1 Member Contribution Assessment

ID	Name	Contribution (%)	Signature
23127069	Nguyen Minh Khoi	100	
23127359	Vo Tran Quoc Duy	100	
23127367	Ha Thu Hoang	100	
23127455	Truong Cong Thien Phu	100	
23127511	Vo Minh Tuan	100	

2 Conceptual Model

The following Extended Entity Relationships Diagram illustrating the semantic entities within the software.



Note for viewing: For a higher-resolution version of this diagram with better visualisation, please use the following link to view the file directly.

Link:

<https://drive.google.com/file/d/1qNROBVLVp7pgbN8k9XpgABbAOBFuqVp2/view?usp=sharing>

3 Architectural Design

3.1 Architecture Diagram

This section breaks down the system into its primary software components, derived from the project's source code structure.

- **Back-End (Python & Flask)**
 - **db/:** Contains the core database definitions.
 - **schema.sql:** Defines the table structures.
 - **constraints.sql:** Defines the relationships and rules between tables.
 - **seed.sql:** Contains initial data for testing.
 - **src/:** The main application source code.
 - **application.py:** The main Flask application entry point.
 - **config.py:** Configuration settings.
 - **extensions.py:** Manages Flask extensions (e.g., SQLAlchemy, JWT).
 - **app/:** The core application package.
 - **auth/:** Handles authentication logic, including login and password reset.
 - **models/:** Contains the SQLAlchemy Object-Relational Mapping (ORM) models that represent database tables (e.g., *account.py*, *class_.py*, *contract.py*).
 - **routes/:** Defines the API endpoints, grouped by user role (e.g., *manager/*, *staff/*, *teacher/*).
 - **schemas/:** Contains the Marshmallow schemas for data validation and serialisation.

- **Front-End (TypeScript & React)**

- **src/:** The main application source code.
 - **assets/:** Stores all static assets, primarily SVG icons grouped by feature (e.g., *header/*, *sidebar/*).
 - **components/:** Contains reusable React components.
 - **auth/:** Components for authentication forms (e.g., *LoginForm.tsx*).
 - **class/:** Components specific to the class view (e.g., *StudentList.tsx*).
 - **layout/:** High-level structural components (e.g., *Sidebar.tsx*, *Header.tsx*, *Footer.tsx*).
 - **ui/:** Generic, low-level UI elements used throughout the application (e.g., *button.tsx*, *card.tsx*, *input.tsx*).
 - **hooks/:** Custom React hooks for managing state and logic (e.g., *useAuthFlow.ts*).
 - **pages/:** Top-level components that represent full application pages (e.g., *AuthPage.tsx*, *ClassScreen.tsx*, *HomeScreen.tsx*).
 - **services/:** Modules responsible for communicating with the back-end API (e.g., *authService.ts*).

Overall System Architecture

The system is designed using a **Three-Tier Client-Server Architecture**. This is a standard and robust model for modern web applications that separates concerns into logical layers.

1. **Presentation Tier (Client):** This is the **front-end**, a single-page application (SPA) built with ReactJS. It runs entirely in the user's web browser. Its main responsibility is to render the user interface and handle all user interactions. It communicates with the Application Tier via a RESTful API.
2. **Application Tier (Server):** This is the **back-end**, built with Python and the Flask framework. It is responsible for handling all business logic, processing data, and managing user authentication and authorisation. It exposes a set of API endpoints that the front-end consumes.
3. **Data Tier (Database):** This tier consists of a **MySQL database**. Its sole responsibility is the persistent storage and retrieval of all application data, with data integrity enforced by a set of schema constraints.

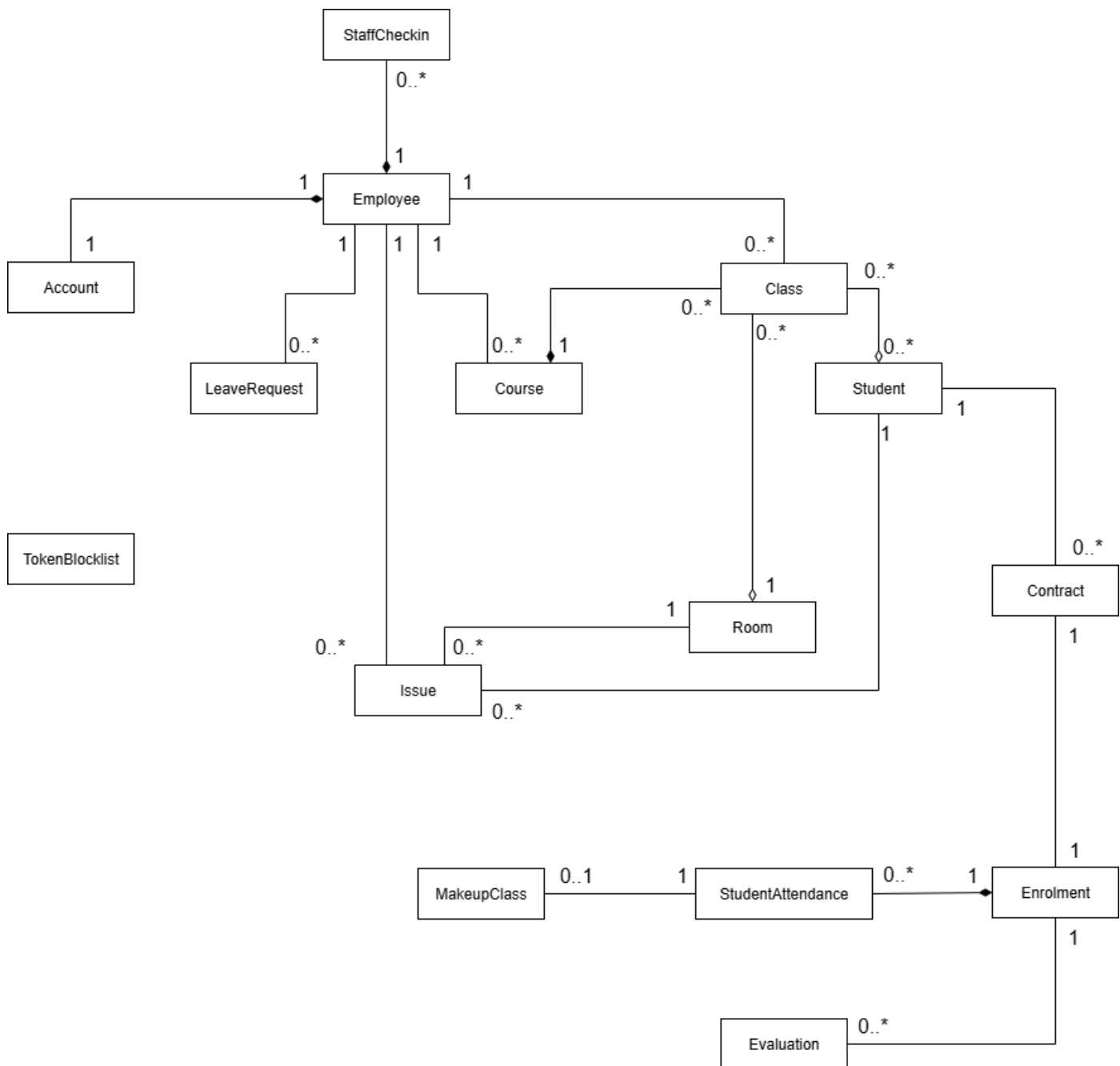
The relationship between these tiers is linear: the **Client** sends HTTP requests to the **Server**, which then processes the request (interacting with the **Database** if necessary) and returns an HTTP response (typically with JSON data) to the Client.

Special Architectural Aspects

Several specific design patterns and architectural styles have been applied to ensure the system is maintainable, scalable, and robust.

- **RESTful API:** Communication between the front-end and back-end is handled exclusively through a RESTful API. This decouples the client from the server, allowing them to be developed and deployed independently.
- **Component-Based Architecture (Front-End):** The React front-end is built using a component-based model. The UI is broken down into small, reusable, and self-contained components (*ui/*, *components/*), which are then composed together to build complex pages (*pages/*).
- **Model-View-Controller (MVC) like Pattern (Back-End):** The Flask back-end is organised in a pattern similar to MVC:
 - **Models:** The SQLAlchemy classes in the *models/* directory represent the data model.
 - **Views:** The "view" is the JSON data serialized by the Marshmallow schemas in the *schemas/* directory.
 - **Controllers:** The functions within the *routes/* directory act as controllers, handling incoming HTTP requests, interacting with the models, and returning the serialized views.
- **Object-Relational Mapping (ORM):** The back-end uses SQLAlchemy as an ORM to map between the Python objects defined in the *models/* directory and the tables in the MySQL database. This abstracts away direct SQL queries and helps prevent vulnerabilities like SQL injection.

3.2 Class Diagram



Note for viewing: For a higher-resolution version of this diagram with better visualisation, please use the following link to view the file directly.

Link:

https://drive.google.com/file/d/1mImPvQV_tDLOMN6R5zoabI_A2XtAYDka/view?usp=sharing

3.3 Class Specifications

This section presents the detailed specifications for the most critical classes in the project. The constraints for each attribute are a synthesis of the database model rules and the API-level validation rules defined in the schemas.

3.3.1 Class *Employee*

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Represents a staff member at the tutoring centre.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key, Required	The unique identifier for an employee.
2	full_name	public	String(2000), Not Null, Required	The employee's full name.
3	email	public	String(320), Not Null, Unique, Required	The employee's unique email address
4	role	public	String(20), Not Null, Check, Required	The employee's role ('Teacher', 'Learning Advisor', 'Manager').
5	phone_number	public	String(20), Optional, Unique	The employee's unique phone number.
6	teacher_status	public	String(20), Optional, Check	The current working status of a teacher ('Available', 'Unavailable').

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	<code>__init__()</code>	public	N/A	The constructor to create an <i>Employee</i> instance.
2	<code>add_employee()</code>	public	N/A	The operation to create a new employee.

3	get_all()	public	N/A	The operation to retrieve all employees.
---	-----------	--------	-----	--

3.3.2 Class: Account

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Stores the login credentials for an *Employee*.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key, Required	The unique identifier for an account.
2	employee_id	public	String(10), Not Null, Unique, Foreign Key, Required	Links to the corresponding <i>Employee</i> .
3	username	public	String(200), Not Null, Unique, Required	The user's unique login name.
4	password	public	String, Required (for input schema)	The plain-text password for creation/login. Not stored in the database.
5	password_hash	public	String(200), Not Null	The user's securely hashed password. Stored in the database.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>Account</i> instance.
2	add_account()	public	Manager Role Required	Related operation in <i>account_route.py</i> to create a new account.
3	login()	public	N/A	Related operation in <i>auth.py</i> to authenticate the user.
4	logout()	public	JWT Required	Related operation in <i>auth.py</i> to invalidate a user's session token.

5	refresh_token()	public	Refresh Token Required	Issues a new access token using a valid refresh token.
6	get_all()	public	N/A	Retrieve all user accounts.
7	get_account()	public	N/A	Retrieves a single account by its ID.

3.3.3 Class: Student

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Represents a student enrolled at the centre.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key	The unique identifier for a student.
2	full_name	public	String(2000), Not Null	The student's full name.
3	contact_info	public	String(200), Not Null	Contact details for the student or their guardian.
4	date_of_birth	public	Date, Not Null	The student's date of birth.
5	created_date	public	Date, Not Null, Default <i>curdate()</i>	The date the student record was created.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>Student</i> instance.

3.3.4 Class: Course

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Stores details about the courses offered.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key	The unique identifier for a course.
2	created_date	public	Date, Primary Key	The date the course record was created.
3	name	public	String(200), Not Null	The name of the course.
4	description	public	String(200), Optional	A brief description of the course.
5	duration	public	Integer, Not Null	The duration of the course in months.
6	start_date	public	Date, Not Null	The official start date of the course.
7	end_date	public	Date, Computed	Automatically calculated based on <i>start_date</i> and <i>duration</i> .
8	schedule	public	String(200), Not Null	The class schedule.
9	learning_advisor_id	public	String(10), Not Null, Foreign Key	Links to the responsible <i>Employee</i> .
10	fee	public	Integer, Not Null	The tuition fee for the course.
11	prerequisites	public	String(20), Not Null	Any prerequisites for enrolling in the course.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	<code>__init__()</code>	public	N/A	The constructor to create an <i>Course</i> instance.

3.3.5 Class: *Class*

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Represents a specific, scheduled instance of a *Course*.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key	The unique identifier for a class.
2	course_id	public	String(10), Primary Key, Foreign Key	Links to the parent <i>Course</i> .
3	course_date	public	Date, Primary Key, Foreign Key	Links to the parent <i>Course</i> .
4	term	public	Integer, Primary Key, Foreign Key	Links to the parent <i>Course</i> .
5	teacher_id	public	String(10), Not Null, Foreign Key	Links to the assigned <i>Employee(Teacher)</i> .
6	room_id	public	String(10), Not Null, Foreign Key	Links to the assigned Room.
7	class_date	public	DateTime, Not Null	The specific date and time of the class session.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>Course</i> instance.

3.3.6 Class: Contract

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Represents a student's enrolment agreement.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key, Required	The unique identifier for a contract.
2	student_id	public	String(10), Not Null, Foreign Key, Required	Links to the <i>Student</i> .
3	employee_id	public	String(10), Not Null, Foreign Key	Links to the <i>Employee</i> who created the contract.

4	course_id	public	String(10), Not Null, Foreign Key, Required	Links to the <i>Course</i> .
5	course_date	public	Date, Not Null, Foreign Key, Required	The creation date of the course instance.
6	tuition_fee	public	Integer, Not Null, Required	The agreed tuition fee for this contract.
7	payment_status	public	String(20), Not Null, Check('In Progress', 'Paid')	The status of the payment.
8	start_date	public	Date, Not Null, Required	The start date of the study period.
9	end_date	public	Date, Not Null, Required	The end date of the study period.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	<code>__init__()</code>	public	N/A	The constructor to create an <i>Contract</i> instance.
2	<code>add_contract()</code>	public	Learning Advisor Role Required	Related operation in <code>contract_route.py</code> to create a new contract.
3	<code>update_contract()</code>	public	Learning Advisor Role Required	Related operation in <code>contract_route.py</code> to update a contract.
4	<code>delete_contract()</code>	public	Learning Advisor Role Required	Related operation in <code>contract_route.py</code> to delete a contract.

3.3.7 Class: *Evaluation*

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Stores grades and comments for student assessments.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
-----	----------	----------	------------	-------------

1	student_id	public	String(10), Composite PK, Foreign Key, Required, Length(10)	Links to the Student.
2	course_id	public	String(10), Composite PK, Foreign Key, Required, Length(10)	Links to the Course.
3	course_date	public	Date, Composite PK, Foreign Key, Required	The creation date of the course instance
4	assessment_type	public	String(200), Composite PK, Check, Required, Check	The type of assessment.
5	teacher_id	public	String(10), Not Null, Foreign Key, Required	Links to the Employee(Teacher) who gave the evaluation.
6	grade	public	String(2), Not Null, Required, Length(2)	The grade received
7	comment	public	String(2000), Not Null, Required	The teacher's written comments
8	enrolment_id	public	String(10), Not Null, Foreign Key, Required	Links to the Enrolment.
9	evaluation_date	public	Date, Not Null, Required	The date the evaluation was made.

- Operations:**

Seq	Operation	Modifier	Constraint	Description
1	<code>__init__()</code>	public	N/A	The constructor to create an <i>Evaluation</i> instance.
2	<code>add_evaluation()</code>	public	Teacher Role Required	Related operation in <code>evaluation_route.py</code> to create an evaluation.
3	<code>update_evaluation()</code>	public	Teacher Role Required	Related operation in <code>evaluation_route.py</code> to update an evaluation
4	<code>delete_evaluation()</code>	public	Teacher Role Required	Related operation in <code>evaluation_route.py</code> to delete an evaluation.

3.3.8 Class: *StaffCheckin*

- **Inheritance:** Inherits from *db.Model*
- **Description:** Stores grades and comments for student assessments.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key, Required	The unique identifier for a check-in record.
2	employee_id	public	String(10), Not Null, Foreign Key, Required	Links to the Employee
3	status	public	String(200), Not Null, Check, Required	The check-in status ('Not Checked In', 'Checked In', 'Late').
4	checkin_time	public	DateTime, Optional, Required (in schema)	The date and time the employee checked in.
5	checkout_time	public	DateTime, Optional, Required (in schema)	The date and time the employee checked out.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>StaffCheckin</i> instance.
2	checkin()	public	N/A	Related operation in <i>checkin_route.py</i> to perform a check-in.
3	checkout()	public	N/A	Related operation in <i>checkin_route.py</i> to perform a check-out.

3.3.9 Class: *StudentAttendance*

- **Inheritance:** Inherits from *db.Model*.
- **Description:** Tracks the attendance of a student for a specific class session.
- **Attributes:**

Seq	Property	Modifier	Constraint	Description
-----	----------	----------	------------	-------------

		er		
1	student_id	public	String(10), Composite PK, Foreign Key, Required	Links to the Student.
2	class_id	public	String(10), Composite PK, Foreign Key, Required	Links to the Class
3	course_id	public	String(10), Composite PK, Foreign Key, Required	Links to the Course
4	course_date	public	Date, Composite PK, Foreign Key, Required	The creation date of the course instance
5	term	public	Integer, Composite PK, Foreign Key, Required	The class term
6	enrolment_id	public	String(10), Not Null, Foreign Key, Required	Links to the Enrolment
7	status	public	String(20), Not Null, Check, Required	The attendance status ('Present', 'Absent')

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>Student Attendance</i> instance.
2	mark_attendance()	public	Teacher Role Required	Marks attendance for a student on a specific date.
3	view_attendance()	public	Teacher, Learning Advisor Role Required	Views attendance records for a student.

3.3.10 Class: Enrolment

- **Inheritance:** Inherits from *db.Model*
- **Description:** An association class that links a student's Contract to a specific Course. It represents the official record of a student being enrolled in a course.

- **Attributes:**

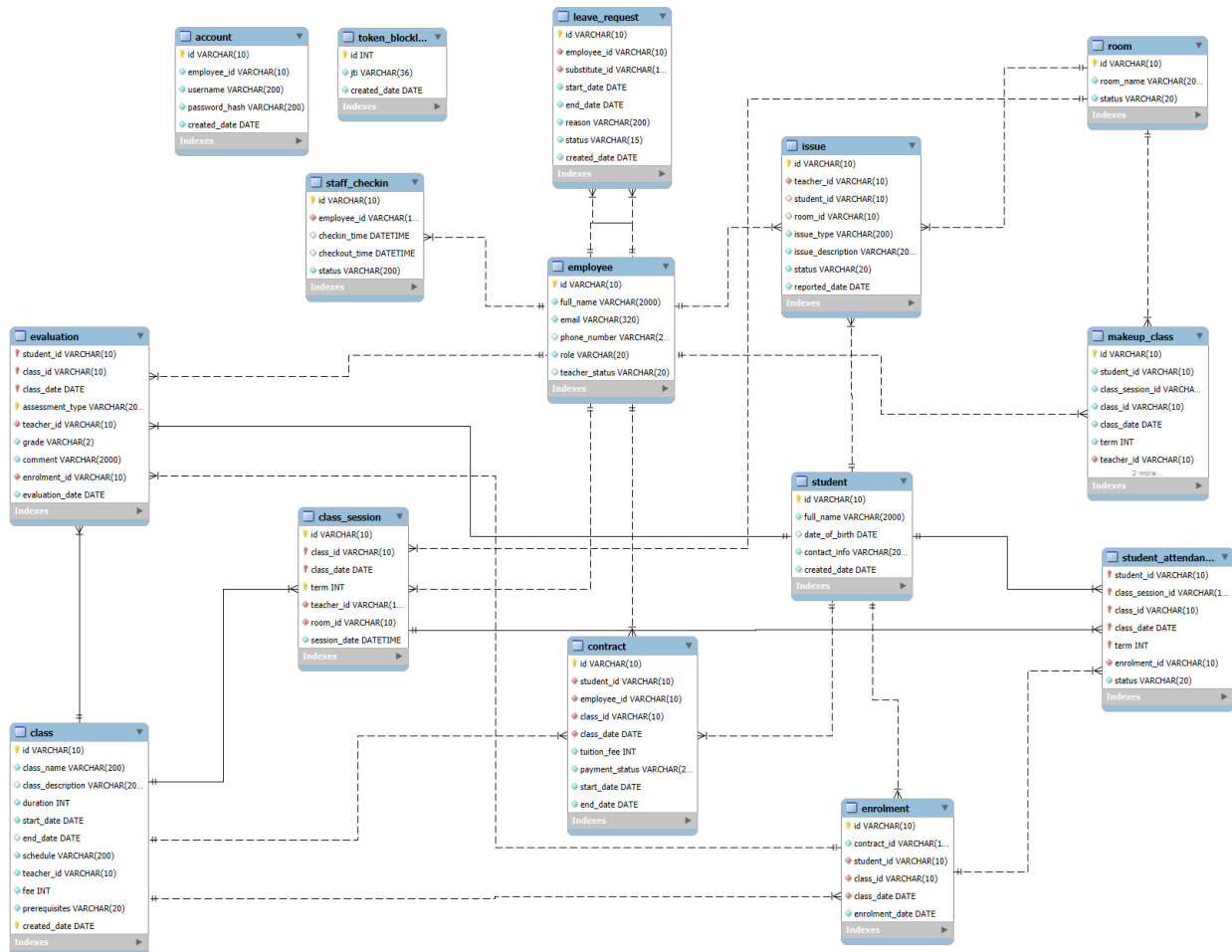
Seq	Property	Modifier	Constraint	Description
1	id	public	String(10), Primary Key	The unique identifier for an enrolment record.
2	contract_id	public	String(10), Not Null, Foreign Key	Links to the Contract that initiated this enrolment.
3	student_id	public	String(10), Not Null, Foreign Key	Links to the Student being enrolled.
4	course_id	public	String(10), Not Null, Foreign Key	Links to the Course the student is enrolled in.
5	course_date	public	Date, Not Null, Foreign Key	The creation date of the course instance, part of the composite key to Course.
6	enrolment_date	public	Date, Not Null	The date the student was officially enrolled.

- **Operations:**

Seq	Operation	Modifier	Constraint	Description
1	__init__()	public	N/A	The constructor to create an <i>Enrolment</i> instance.

4 Data Design

4.1 Data Diagram



4.2 Data Specification

The following tables describe the database schema for the Tutoring Centre Management System. Each table includes a breakdown of its attributes, data types, and constraints.

- **Table:** employee

Purpose: Stores information about all staff members at the centre.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for an employee.
full_name	NVARCHAR(2000)	NOT NULL: The employee's full name.
email	VARCHAR(320)	NOT NULL, UNIQUE: The employee's unique email address.
phone_number	VARCHAR(20)	UNIQUE: The employee's unique phone number
role	VARCHAR(20)	NOT NULL, CHECK: The employee's role. Must be one of 'Teacher', 'Learning Advisor', or 'Manager'.
teacher_status	VARCHAR(20)	CHECK: Only applicable if <i>role</i> is 'Teacher'. Must be 'Available' or 'Unavailable'.

- **Table:** staff_checkin

Purpose: Tracks daily check-in and check-out times for staff.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for a check-in record.
employee_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> .

checkin_time	DATETIME	The date and time the employee checked in.
checkout_time	DATETIME	The date and time the employee checked out. <i>checkout_time</i> must be after <i>checkin_time</i> .

- **Table:** leave_request

Purpose: Stores leave request submitted by employees.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for a leave request.
employee_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> . The employee requesting leave.
substitute_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> . The employee designated as the substitute. Cannot be the same as <i>employee_id</i> .
start_date	DATE	NOT NULL: The start date of the leave. Must be before <i>end_date</i> .
end_date	DATE	NOT NULL: The end date of the leave.
reason	NVARCHAR(200)	NOT NULL: The reason for the leave request.
status	NVARCHAR(15)	NOT NULL, CHECK, DEFAULT 'Not Approved': The approval status. Must be 'Approved' or 'Not Approved'.
created_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the request was created.

- **Table:** account

Purpose: Stores user account and credential information for employees.

Attribute Name	Data Type	Constraints and Descriptions
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for an account.
employee_id	VARCHAR(10)	NOT NULL, UNIQUE, FOREIGN KEY: References <i>employee(id)</i> . Creates a one-to-one relationship.
username	VARCHAR(200)	NOT NULL, UNIQUE: The user's unique login name.
password_hash	VARCHAR(200)	NOT NULL: The user's securely hashed password.
created_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the account was created.

- **Table:** student

Purpose: Stores personal information for each student.

Attribute Name	Data Type	Constraint and Descriptions
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for a student.
full_name	NVARCHAR(2000)	NOT NULL: The student's full name.
date_of_birth	DATE	NOT NULL: The student's date of birth.
contact_info	VARCHAR(200)	NOT NULL: Contact details for the student or their guardian.
created_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the student record was created.

- **Table:** course

Purpose: stores details about the courses offered by the centre.

Attribute Name	Data Type	Constraints and Descriptions
id	VARCHAR(10)	PRIMARY KEY (Composite with <i>created_date</i>): The unique identifier for a course.
name	VARCHAR(200)	NOT NULL: The name of the course.
description	VARCHAR(200)	A brief description of the course.
duration	INT	NOT NULL: The duration of the course in months.
start_date	DATE	NOT NULL: The official start date of the course.
end_date	DATE	GENERATED: Automatically calculated as <i>start_date</i> plus the <i>duration</i> in months.
schedule	VARCHAR(200)	NOT NULL: The schedule for the course (e.g., "Mon - Wed 9:00 - 10:30").
learning_advisor_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> . The staff member responsible for the course.
fee	INT	NOT NULL: The tuition fee for the course.
prerequisites	VARCHAR(200)	NOT NULL: Any prerequisites for enrolling in the course.
created_date	DATE	PRIMARY KEY (Composite with id): The date the course record was created.

- **Table:** enrolment

Purpose: Links a student's contract to a specific course instance.

Attribute Name	Data Type	Constraints and Descriptions
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for an enrolment.
contract_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>contract(id)</i> .
student_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>student(id)</i> .
course_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: Composite key referencing <i>course(id, created_date)</i> .
course_date	DATE	NOT NULL, FOREIGN KEY: Composite key referencing <i>course(id, created_date)</i> .
enrolment_date	DATE	NOT NULL: The date the student was officially enrolled.

- **Table:** room

Purpose: Stores information about the classrooms.

Attribute Name	Data Type	Constraints and Descriptions
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for a room.
room_name	VARCHAR(200)	NOT NULL: The name or number of the room.
status	VARCHAR(20)	NOT NULL, CHECK, DEFAULT 'Free': The current status of the room. Must be 'Free', 'Occupied', or 'Maintenance'.

- **Table:** class

Purpose: Represents a specific instance of a course taught by a teacher in a room.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY (Composite): The unique identifier for a class.
course_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>course(id, created_date)</i> .
course_date	DATE	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>course(id, created_date)</i> .
term	INT	PRIMARY KEY (Composite), CHECK: The term of the class. Must be 1 or 2.
teacher_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> . The teacher for this class.
room_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>room(id)</i> . The room for this class.
class_date	DATETIME	NOT NULL: The specific date and time of the class session.

- **Table:** student_attendance

Purpose: Tracks the attendance of a student for a specific class session.

Attribute Name	Data Type	Constraints and Description
student_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: References <i>student(id)</i> .

class_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>class</i> .
course_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>class</i> .
course_date	DATE	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>class</i> .
term	INT	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>class</i> .
enrolment_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>enrolment(id)</i> .
status	VARCHAR(20)	NOT NULL, CHECK: The attendance status. Must be 'Present' or 'Absent'.

- **Table:** makeup_class

Purpose: Stores information about a scheduled make-up class for a student's absence.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for the make-up class record.
student_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: Composite key referencing a specific <i>student_attendance</i> record.
class_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: Composite key referencing <i>student_attendance</i> .

course_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: Composite key referencing <i>student_attendance</i> .
course_date	DATE	NOT NULL, FOREIGN KEY: Composite key referencing <i>student_attendance</i> .
term	INT	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> . The teacher for the make-up class.
room_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>room(id)</i> . The room for the make-up class.
created_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the make-up class was scheduled.

- **Table:** evaluation

Purpose: Stores grades and comments from teachers for student assessments.

Attribute Name	Data Type	Constraints and Description
student_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: References <i>student(id)</i> .
course_id	VARCHAR(10)	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>course</i> .
course_date	DATE	PRIMARY KEY (Composite), FOREIGN KEY: Composite key referencing <i>course</i> .
assessment_type	VARCHAR(200)	PRIMARY KEY (Composite), CHECK: The type of assessment (e.g., 'Quiz 1', 'Writing Project 1').
teacher_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> .

grade	VARCHAR(2)	NOT NULL: The grade received.
comment	VARCHAR(2000)	NOT NULL: The teacher's written comments.
enrolment_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>enrolment(id)</i> .
evaluation_date	DATE	NOT NULL: The date the evaluation was made.

- **Table:** issue

Purpose: Tracks issues reported by teachers.

Attribute Name	Data Type	Constraints and Description
id	VARCHAR(10)	PRIMARY KEY: The unique identifier for an issue.
teacher_id	VARCHAR(10)	NOT NULL, FOREIGN KEY: References <i>employee(id)</i> .
student_id	VARCHAR(10)	FOREIGN KEY: References <i>student(id)</i> . Only populated if issue_type is 'student Behavior'.
room_id	VARCHAR(10)	FOREIGN KEY: References <i>room(id)</i> . Only populated if issue_type is 'Technical'.
issue_type	VARCHAR(200)	NOT NULL, CHECK: The type of issue. Must be 'student Behavior' or 'Technical'.
issue_description	VARCHAR(200)	NOT NULL: A description of the issue.
status	VARCHAR(20)	NOT NULL, CHECK, DEFAULT 'In Progress': The status of the issue. Must be 'In Progress' or 'Done'.

reported_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the issue was reported.
---------------	------	---

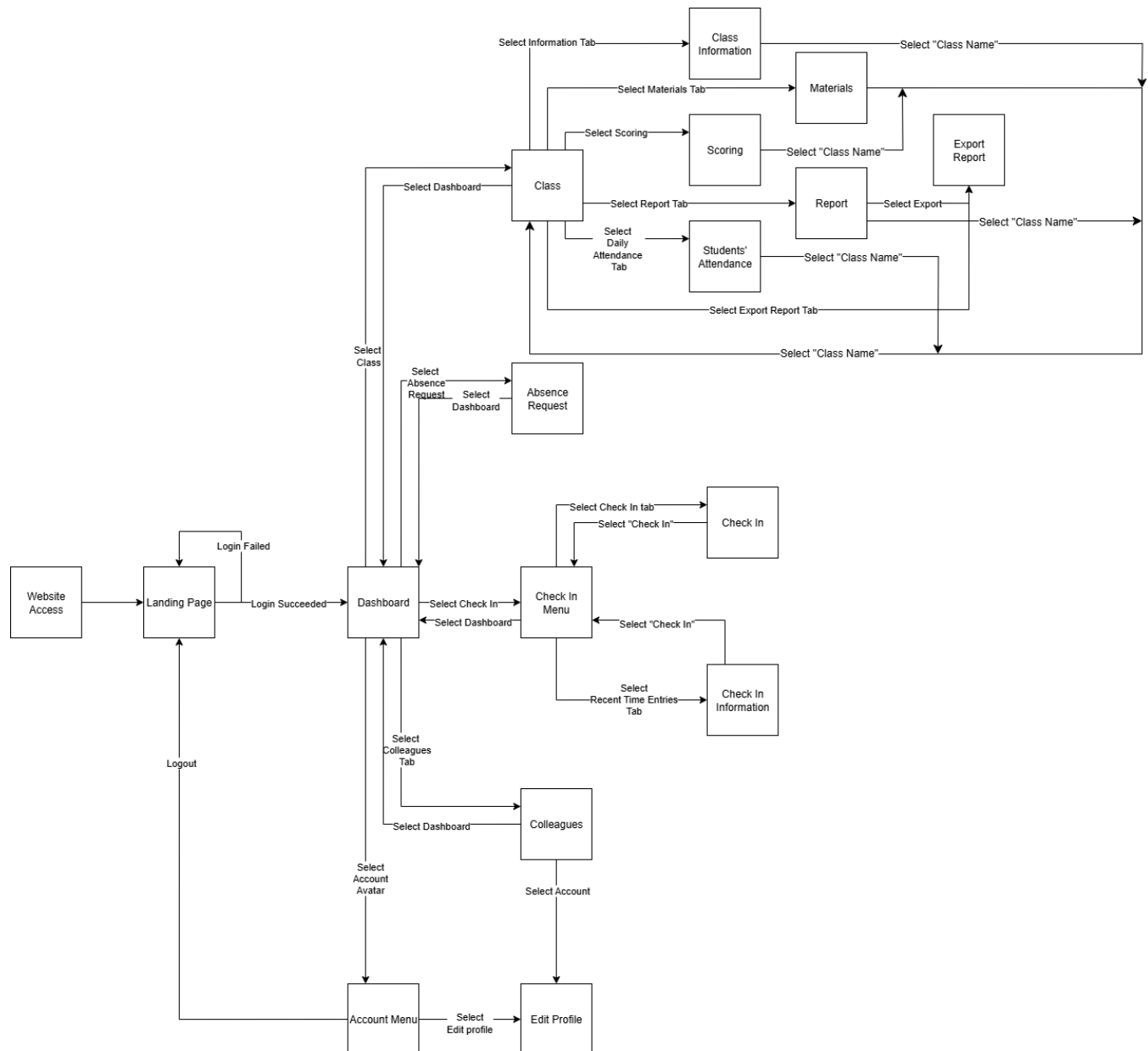
- **Table:** token_blocklist

Purpose: Stores invalidated JWTs to handle user logouts.

Attribute Name	Data Type	Constraints and Description
id	INT	PRIMARY KEY, AUTO_INCREMENT: A unique integer ID for the blocklist record.
jti	VARCHAR(36)	NOT NULL, UNIQUE: The unique identifier of the blocklisted JWT.
created_date	DATE	NOT NULL, DEFAULT CURRENT_DATE: The date the token was blocklisted.

5 User Interface and User Experience Design

5.1 Screen Diagram

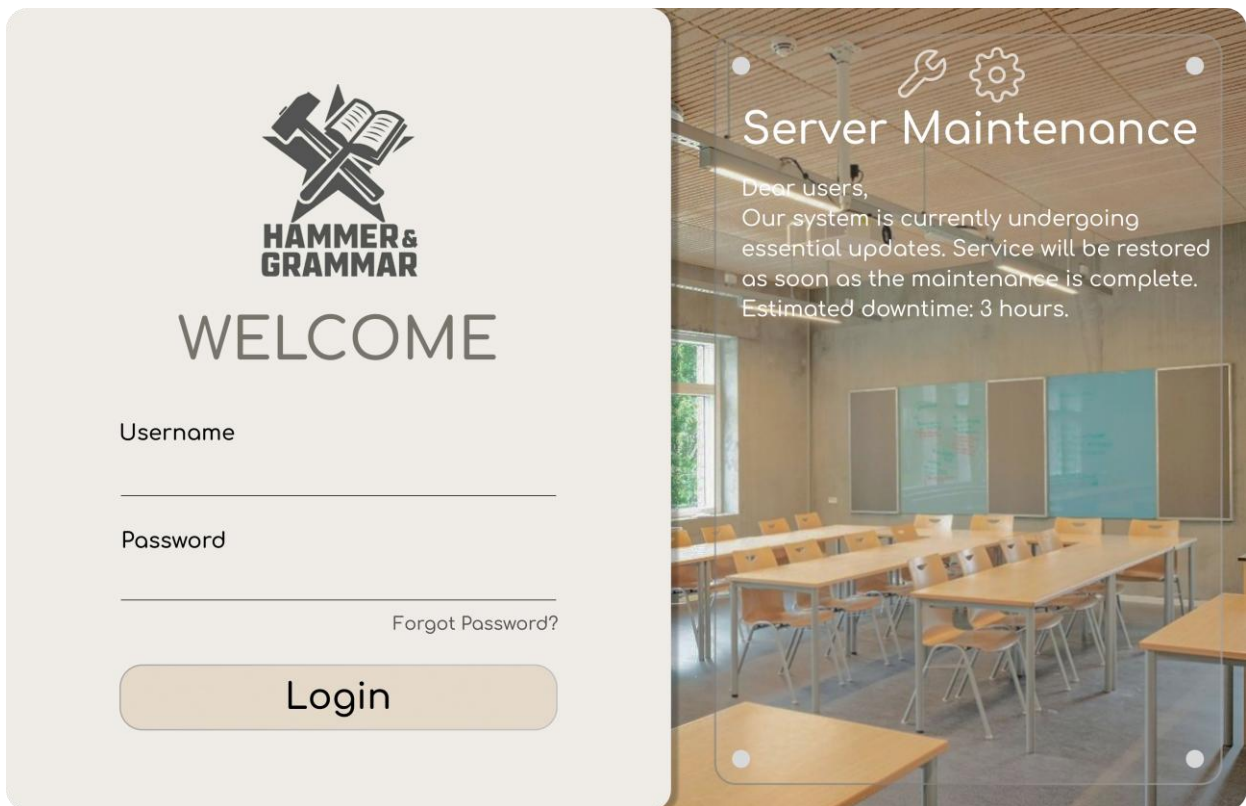


Seq	Screen	Description
1	Landing Page	The initial application entry point, providing a secure login form and displaying system-wide notices like server maintenance.
2	Dashboard	The main screen after login, displaying a list of the user's classes with progress bars, status filters, and a search function.
3	Class Screen	The default tab within a class, showing a detailed list of enrolled students and their contact information.
4	Class Information Screen	Provide general information about the class.
5	Class Scoring Screen	The interface for Teachers to input detailed grades for students across various assessments (e.g., Q1, Q2).
6	Class Attendance Screen	The interface for Teachers to take daily attendance for a selected class session, marking each student's presence
7	Class Materials Screen	A view for managing and accessing class-related documents and materials, with options to add, download, or delete files.
8	Class Report Screen	A screen that consolidates each student's performance within a class, including scores and a written teacher assessment.
9	Check In Screen	The primary screen for staff timekeeping, featuring a prominent check-in button, monthly stats, and a list of upcoming classes.
10	Absence Request Screen	A clean, form-based screen for Teachers to

		submit applications for temporary absence.
11	Recent Time Entries Screen	A view showing a user's historical check-in records with their status ("On time" or "Late").
12	Colleagues Screen	A staff directory that lists all colleagues and displays a detailed professional profile for a selected person.
13	Profile Setting Screen	An interface for users to view and edit their own professional profile information, including their nickname, philosophy, and achievements.

5.2 Screen Specifications

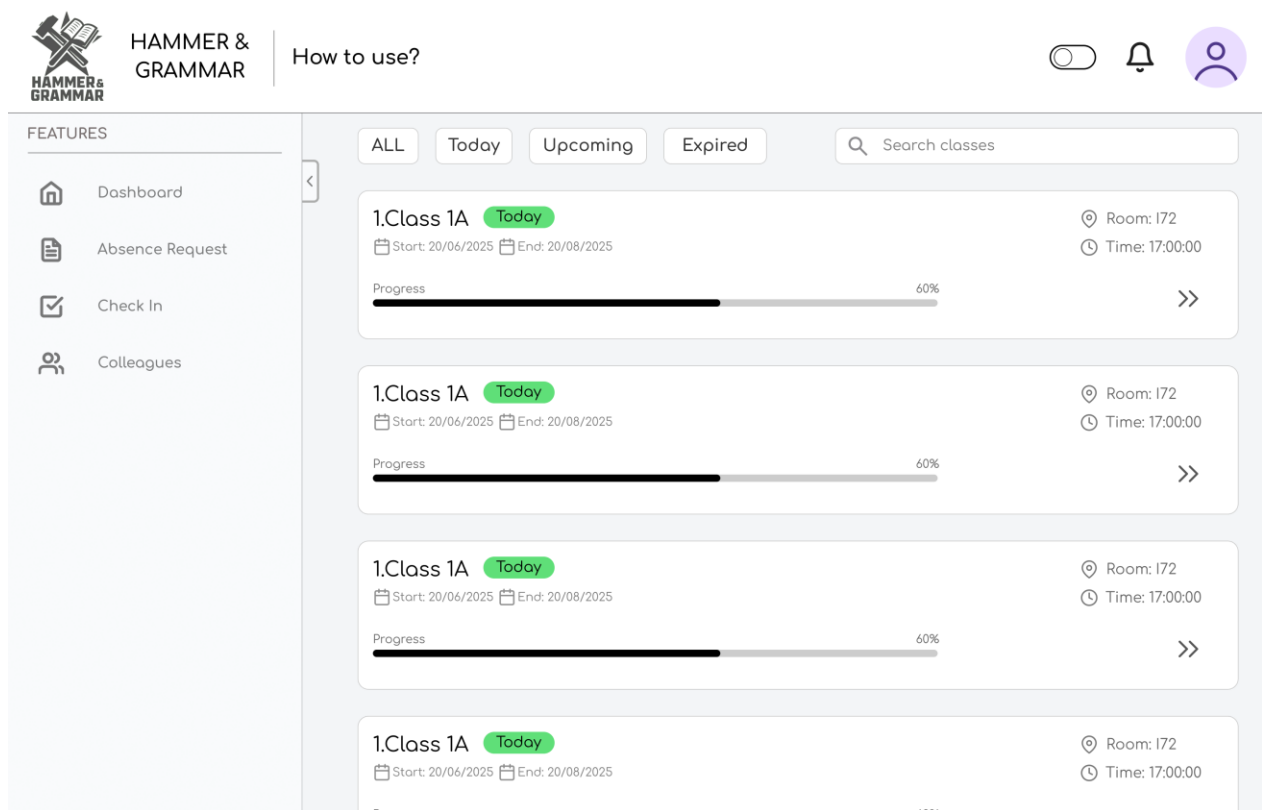
5.2.1 Screen "Landing Page"



- **Presentation Format:**

- The screen features a two-panel layout with a prominent login form on the left and a background image with a Notification on the right.
- **Login Form:** Contains fields for "Username" and "Password", a "Forgot Password?" link, and a large "Login" button.
- **Header:** A minimal header displays the "HAMMER & GRAMMAR" logo.
- **Notice Panel:** A semi-transparent panel on the right displays important system-wide messages.

5.2.2 Screen "Dashboard"



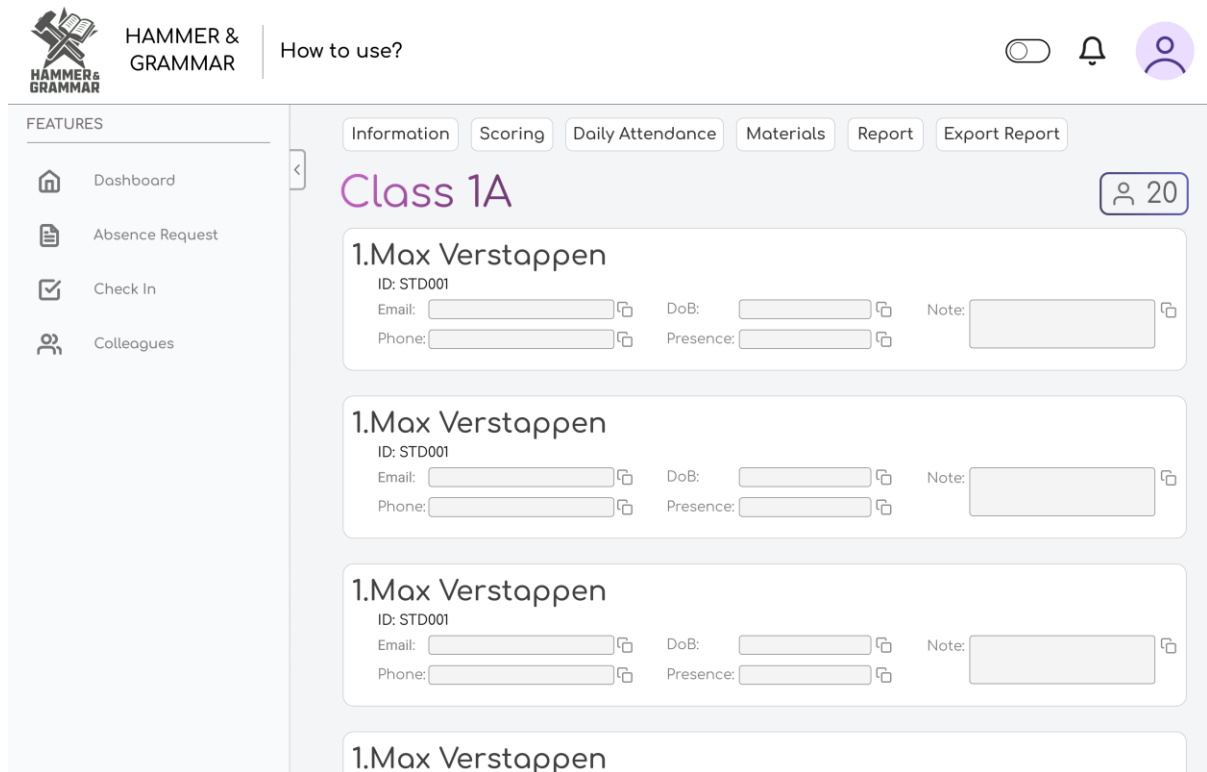
This screen serves as the central hub for authenticated users, providing an overview of their classes and access to all main features of the application.

- **Presentation Format:**

The screen is composed of a standard header, a collapsible sidebar, and a main content area.

- **Header:** Contains the
 - **HAMMER & GRAMMAR** logo.
 - A set of controls on the right, including a **mode toggle switch**, a **notification icon**, and a **user profile icon**.
- **Sidebar:**
 - A vertical navigation bar titled "**FEATURES**".
 - Contains navigation links with icons and text for **Dashboard**, **Absence Request**, **Check In**, and **Colleagues**.
- **Main Content:**
 - **Filter Bar:** A horizontal bar at the top with filter buttons: "**ALL**", "**Today**", "**Upcoming**", and "**Expired**". A "**Search classes**" input field is also present.
 - **Class List:** A scrollable list of class cards. Each card displays:
 - Class name (e.g., "1.Class 1A") and a status tag (e.g., "Today").
 - **Start** and **End** dates.
 - A visual **Progress** bar with a percentage.
 - **Room** number and class **Time**.
 - A navigation arrow (>>) to view class details.
- **Event Handling:**
 - **on Click (Sidebar Link):** Navigates the user to the corresponding screen (e.g., clicking "**Absence Request**" opens the leave request form)
 - **on Click (Filter Button):** The class list is filtered to show only classes matching the selected status (e.g., "**Today**").
 - **on Click (Class Card or >> arrow):** Navigates the user to the detailed screen for that specific class
 - **on Click (User Profile Icon):** A dropdown menu appears with options to "**Edit profile**" or "**Log Out**" of the application.

5.2.3 Screen "Class"



This specification describes the Class Screen, which serves as the detailed hub for all information and activities related to a single class. The screen uses a tabbed interface to organise its extensive functionality.

- **Presentation Format**

The screen maintains the application's standard layout, including the main header and collapsible "FEATURES" sidebar. The main content area is titled with the class name (e.g., "Class 1A") and features a horizontal tab bar for navigation between its different sections.

- **Tab Bar:**

- A list of navigation tabs is displayed at the top of the content area: **Information, Scoring, Daily Attendance, Materials, Report, and Export Report.**

- **"Information" Tab:**

- This is the default view. It displays a scrollable list of all enrolled students.
 - Each student is listed in a panel showing their Name and ID, along with fields for Email, Phone, DoB, Presence, and a

Note.

- When selected, an overall information of the class will be shown.
- **"Scoring" Tab:**
 - This tab is for grading. It displays a list of students with a header row defining assessment columns (e.g., Q1, Q2, Q3, Q4, Q5, OVERALL).
 - Each student row contains corresponding input fields for their scores.
 - A "Select" dropdown allows the teacher to filter by assessment type (e.g., Homework, Midterm, Final).
- **"Daily Attendance" Tab:**
 - This tab is for marking attendance. It shows a list of students for a selected Date.
 - Each student row has a checkbox on the right to mark their presence.
 - A "SAVE CHANGES" button is located at the top to persist the records.
- **"Materials" Tab:**
 - This tab is for managing class materials. It features an "Add Materials" button and a list of uploaded files (e.g., "Chapter 1.pdf").
 - Each file in the list has a download icon and a delete icon.
- **"Report" Tab:**
 - This tab consolidates student performance into a summary report.
 - It lists each student with their scores for major assessments (Homework, Midterm, Final) and a large text area for a written "Teacher Assessment".
 - An "Export" button is available to generate a printable version of the report.
- **Event Handling**
 - **on Click (Navigation Tab):** Switches the main content area to display the selected section (e.g., clicking "Scoring" shows the

- grading grid)
- **on Click ("SAVE CHANGES" button in Attendance):** The system validates and saves the attendance status for all students for the selected date.
- **on Change (Score Input in Scoring):** A teacher can enter a numeric grade for a student. The "OVERALL" score for that student is updated automatically.
- **on Click ("Add Materials" button in Materials):** Opens a file upload dialogue, allowing the teacher to add new documents to the class.
- **on Click ("Export" button in Report):** The system generates a comprehensive report file for the class and initiates a download. Upon successful file generation, a pop-up notification will appear to inform the user that the report was successfully exported.
- **on Change (Score Input in Scoring):** A teacher can enter a numeric grade for a student. The "OVERALL" score for that student is updated automatically.
- **on Click (Download/Delete icon in Materials):** Initiates a file download or prompts the user to confirm deletion of the selected material.
- **on Click (Class Name (e.g. Class 1A)):** Sends the user back to the main class screen.
- **Functional Screen**
 - **Class Information**

HAMMER & GRAMMAR | How to use?

FEATURES

- Dashboard
- Absence Request
- Check In
- Colleagues

Information | **Scoring** | Daily Attendance | Materials | Report | Export Report

Class 1A

20

Start: 20/06/2025 | End: 20/08/2025

Room: 172 | Time: 17:00:00

Progress: 60%

○ Scoring

HAMMER & GRAMMAR | How to use?

FEATURES

- Dashboard
- Absence Request
- Check In
- Colleagues

Information | **Scoring** | Daily Attendance | Materials | Report | Export Report

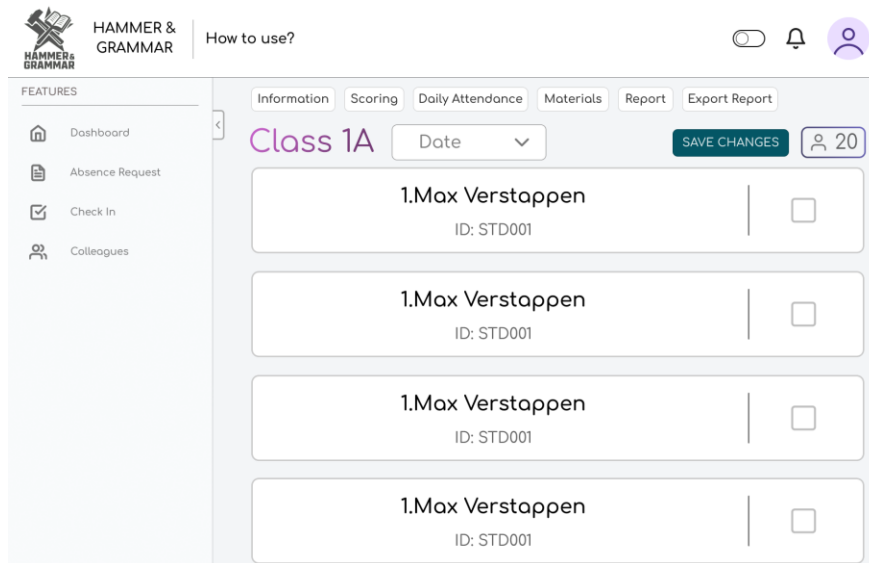
Class 1A

Select

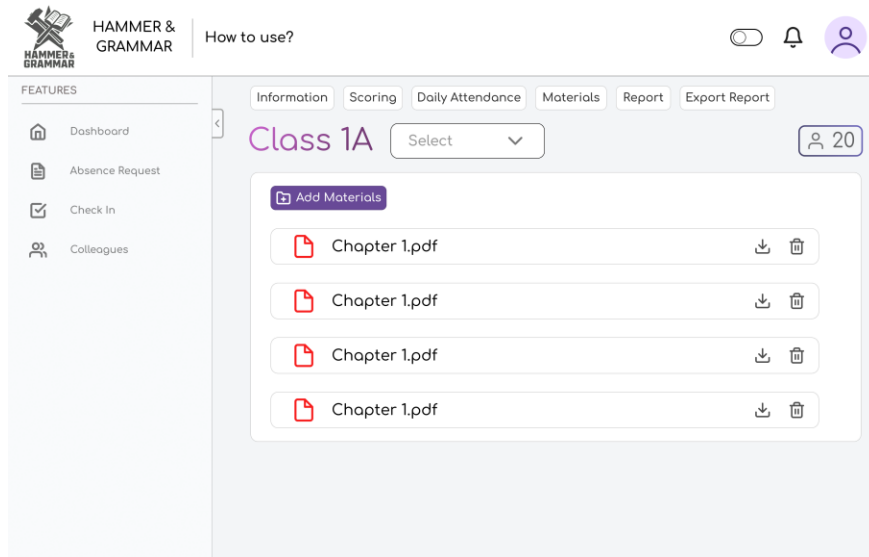
20

NAME	ID	Q1	Q2	Q3	Q4	Q5	OVERALL
Max Verstappen	STD001						
Max Verstappen	STD001						
Max Verstappen	STD001						
Max Verstappen	STD001						

○ Daily Attendance



- **Class Materials**



- **Class Report**

The screenshot shows the 'Report Class 1A' interface. At the top, there's a header with the 'HAMMER & GRAMMAR' logo, a 'How to use?' link, and user controls (toggle, bell, profile). Below the header is a navigation bar with tabs: 'Information', 'Scoring', 'Daily Attendance', 'Materials', and 'Report'. The 'Report' tab is active. On the left, a 'FEATURES' sidebar lists 'Dashboard', 'Absence Request', 'Check In', and 'Colleagues'. The main content area is titled 'Report Class 1A' with an 'Export' button and a user count '20'. It displays a list of student reports for '1.Max Verstappen' (ID: STD001). Each entry includes 'Homework', 'Midterm', and 'Final' scores, and a 'Teacher Assessment' section with a text area for comments.

5.2.4 Screen "Absence Request"

The screenshot shows the 'Absence Request' interface. At the top, there's a header with the 'HAMMER & GRAMMAR' logo, a 'How to use?' link, and user controls (toggle, bell, profile). Below the header is a navigation bar with tabs: 'Information', 'Scoring', 'Daily Attendance', 'Materials', and 'Report'. The 'Report' tab is active. On the left, a 'FEATURES' sidebar lists 'Dashboard', 'Absence Request', 'Check In', and 'Colleagues'. The main content area is titled 'Absence Request' with a document icon. It contains a form with the following fields: 'Type of Absence' (a dropdown menu with 'Select a reason...' and a downward arrow), 'Start Date' (a text input with 'dd/mm/yyyy' and a calendar icon), 'End Date' (a text input with 'dd/mm/yyyy' and a calendar icon), and 'Substitution Plan & Note' (a large text area with a placeholder text: 'e.g., Ms. Jane Doe will cover my classes. All materials are on shared drive'). A purple 'SUBMIT' button is located at the bottom right of the form.

This screen provides the interface for Teachers and other staff to submit a formal request for temporary absence.

- **Presentation Format**

The screen uses the application's standard layout, including the main header and collapsible "FEATURES" sidebar. The main content area is dedicated to the absence request form.

- **Header**

- A large title, "**Absence Request**", is displayed with a document icon to the left.

- **Form Container:**

- The form is presented within a clean, white card with rounded corners.

- **Form Fields:**

- **Type of Absence:** A dropdown menu with the placeholder "Select a reason...".
 - **Start Date / End Date:** Two date input fields with placeholder text dd/mm/yyyy and an interactive calendar icon next to each.
 - **Substitution Plan & Note:** A large text area for detailed notes and arrangements, with placeholder text suggesting examples.

- **Action Button:**

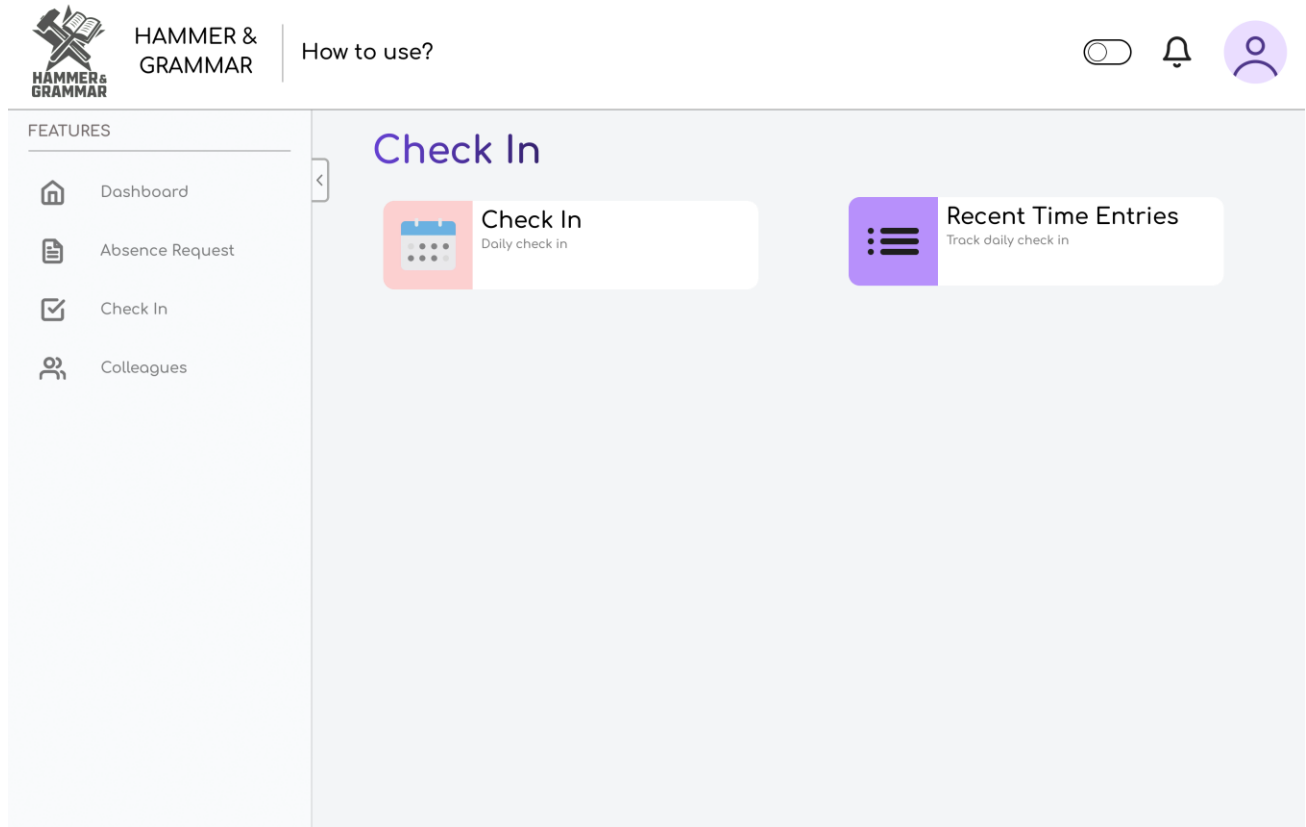
- A prominent purple "**SUBMIT**" button is located at the bottom right of the form. The button has multiple interaction states, including a darker hover/active state and a "DONE!" success state.


- **Event Handling**

- **on Click (Date Input / Calendar Icon):** An interactive calendar widget appears, allowing the user to visually select the start and end dates for their absence.
 - **on Change (Input Fields):** The user fills in the details of their leave request, selecting a reason from the dropdown and typing their plan into the text area.
 - **on Click ("SUBMIT" Button):**
 - The system validates that all required fields are filled. If any are invalid, field-specific error messages are displayed.



- If the data is valid, the request is sent to the server. The button's appearance may change to a darker shade during this process.
- Upon successful submission, the button's text briefly changes to "DONE!" to provide clear visual feedback to the user.

5.2.5 Screen "Check In"





HAMMER & GRAMMAR


How to use?


☐  


FEATURES

 Dashboard


 Absence Request


 Check In


 Colleagues



Check In

 **5 Days**
This month



 **05:00 PM**
Avg Time







Today's Check-in
Wednesday, July 23, 2025



Check In



Upcoming Classes


Class 1A  20
 Room: 172
Class will start in 5 minutes...

Class 2B  20
 Room: 172
Class will start in 2 hours...



Class 3D  20
 Room: 172
Class will start in 5 hours...

Class 1B  20
 Room: 172
Class will start in 7 hours...


Class 2F  20
 Room: 172
Class will start in 12 hours...


HAMMER & GRAMMAR


How to use?


☐  

FEATURES

 Dashboard


 Absence Request

 Check In

 Colleagues

Recent Time Entries

Tuesday, July 23, 2025	Clock In 05:30:00 PM	Late
Monday, July 22, 2025	Clock In 05:00:00 PM	On time
Tuesday, July 16, 2025	Clock In 05:30:00 PM	Late
Tuesday, July 09, 2025	Clock In 05:30:00 PM	Late
Tuesday, July 02, 2025	Clock In 05:30:00 PM	Late
Tuesday, June 25, 2025	Clock In 05:30:00 PM	Late



You have been late many days recently, please be on time next time.

This specification describes the user flow and interface for the staff check-in feature, which is composed of three main screens: a central hub, a check-in screen, and a history view.

- **Screen: Timekeeping Hub**

This screen serves as the main entry point for all timekeeping-related activities.

- **Presentation Format**

- The screen is titled "Check In"
 - Two large, clickable cards are presented to the user:
 - **"Check In"**: With a calendar icon and a "Daily check in" subtitle.
 - **"Recent Time Entries"**: With a list icon and a "Track daily check in" subtitle.

- **Event Handling**

- **on Click ("Check In" card)**: Navigates the user to the detailed Check In Screen.
 - **on Click ("Recent Time Entries" card)**: Navigates the user to the Recent Time Entries Screen.

- **Screen: Check In Screen**

This screen is the primary interface for a staff member to perform their daily check-in.

- **Presentation Format**

- **Header**: Titled **"Check In"** with a calendar icon.
 - **Statistics**: Two summary cards display statistics for the user: days worked this month and average check-in time.
 - **Check-in Card**: A large central card for **"Today's Check-in"** features a background image, the current date, and a prominent **"Check In"** button. After being clicked, this button changes to a "CLOCKED IN" state.
 - **Upcoming Classes**: A panel on the right lists the user's upcoming classes for the day, including the class name, room, and time until it starts.

- **Event Handling**

- **on Click ("Check In" Button)**: The system records the user's

attendance for the day. The button's state changes to "CLOCKED IN" to provide immediate visual feedback and prevent duplicate entries. This action fulfills the core of requirement FR-ACC-02.

- **Screen: Recent Time Entries**

This screen allows staff to review their attendance history.

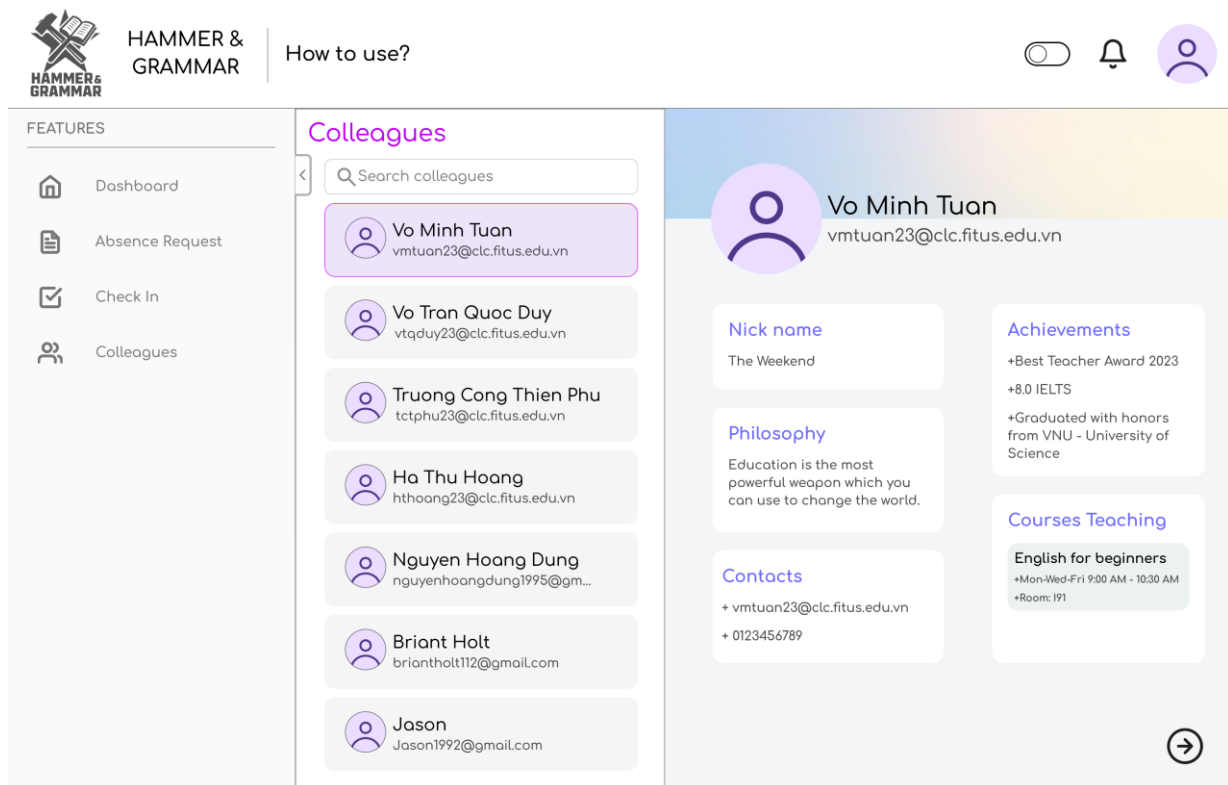
- **Presentation Format**

- **Title:** The screen is titled "**Recent Time Entries**".
- **History List:** A scrollable list displays past check-in records. Each entry shows the Date, the Clock In time, and a status tag (e.g., "On time" or "Late").
- **Warning Message:** A conditional panel on the right displays a warning message with an alert icon if the user has been late frequently.

- **Event Handling**

- This screen is primarily informational. The main interaction is scrolling through the list of past entries.

5.2.6 Screen "Colleagues"



This screen functions as a staff directory, allowing users to view a list of their colleagues and see detailed profile information for each person.

- **Presentation Format**

The screen uses the application's standard layout with a header and sidebar, while the main content area is divided into two vertical panels.

- **Header and Sidebar:**

- The screen includes the standard application header and the collapsible "FEATURES" sidebar.

- **Left Panel (Colleague List):**

- This panel is titled "**Colleagues**" and features a "**Search colleagues**" input field at the top.
 - Below the search bar is a scrollable list of all staff members.
 - Each item in the list displays a colleague's profile icon, full name, and email address. The currently selected colleague is visually highlighted.

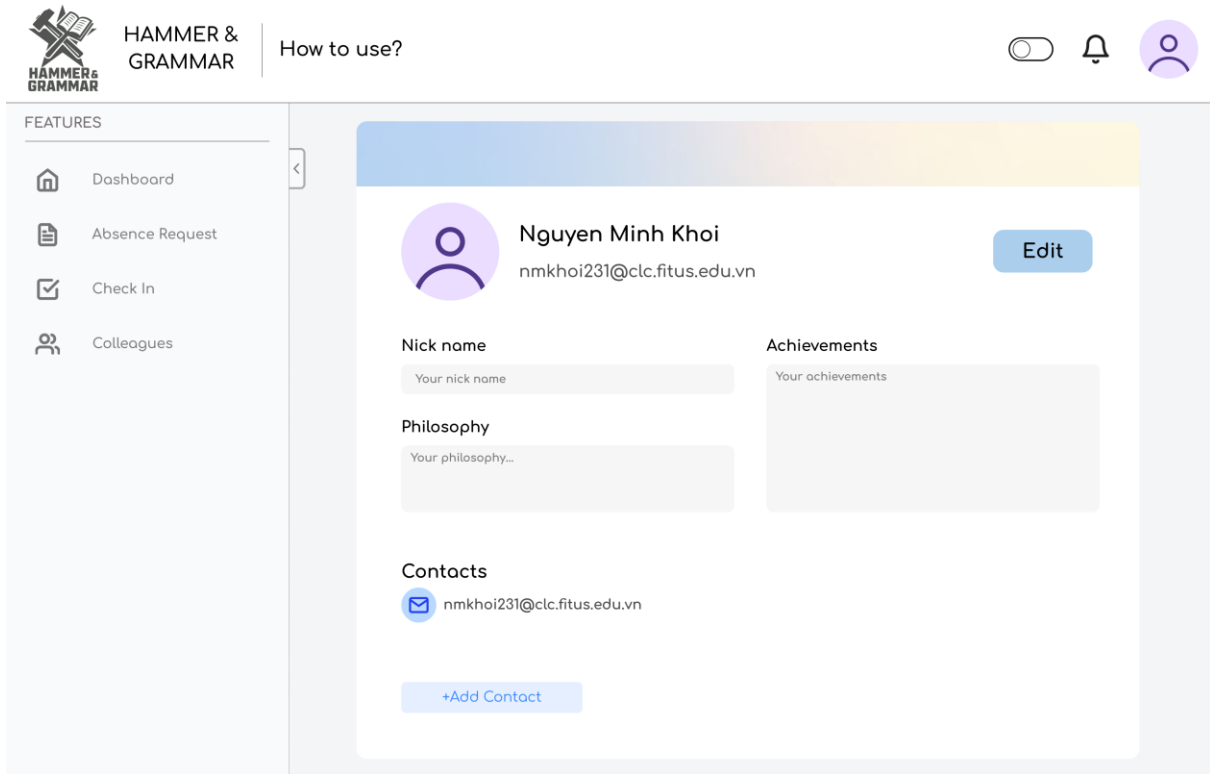
- **Right Panel (Profile Details):**

- This panel displays the detailed profile of the colleague selected from the list on the left.
 - It features a large profile icon, the colleague's full name, and email at the top.
 - The information is organised into several cards: **Nick name, Philosophy, Contacts, Achievements, and Courses Teaching.**

- **Event Handling**

- **on Change (Search colleagues input):** As the user types into the search bar, the list of colleagues in the left panel is filtered in real-time to show only names matching the search term.
 - **on Click (Colleague in List):** When a user clicks on a colleague in the left panel, that item becomes highlighted, and the right panel dynamically updates to display the detailed profile information for that selected person.
 - **Scrolling (Left Panel):** The user can scroll vertically through the list to view all staff members.

5.2.7 Screen "Profile Setting"



This screen allows users to view and edit their own personal and professional profile information.

- **Presentation Format**

The screen uses the application's standard layout with a header and sidebar. The main content area is dedicated to displaying the user's profile.

- **Header and Sidebar:**

- The screen includes the standard application **header** and the collapsible **"FEATURES" sidebar**.

- **Main Content Area:**

- **Profile Header:** A banner at the top displays the user's profile icon, full name (e.g., "Nguyen Minh Khoi"), and email address. A prominent **"Edit"** button is located on the right.
- **Editable Fields:** The profile information is organized into several sections with input fields:
 - **Nick name:** A text field for a user's preferred name.
 - **Philosophy:** A text area for a user's teaching philosophy or personal

motto.

- **Achievements:** A large text area for a user to list their accomplishments.
- **Contacts:** A section displaying existing contact information and an "+Add Contact" button to add more details.
- **Event handling:**
 - **on Click ("Edit" Button):** This action switches the profile fields from a read-only state to an editable state. The "Edit" button would likely change to a "Save" button, allowing the user to commit their changes.
 - **on Change (Input Fields):** When in edit mode, the user can type new information into the "Nick name", "Philosophy", and "Achievements" fields.
 - **on Click ("Add Contact" Button):** When in edit mode, this action would add a new input field to the "Contacts" section, allowing the user to add another line of contact information (e.g., a phone number).
 - **on Click ("Save" Button - implied):** After making changes, the user would click the "Save" button to persist their updated profile information to the database. A success notification would then be displayed.