1 INTRODUCTION

1.1 OVERVIEW

With recent advances in technology and the Internet, the concept of teaching and learning have evolved significantly. Conventional face-to-face teaching is becoming a thing of the past as knowledge is everywhere and accessible from anywhere (Ghosh et al., 2019).

The emergence of e-learning platforms has revolutionized the education sector, and they have become a popular choice among students and professionals seeking to acquire new skills or advance their careers (Naveen et al., 2023). Online learning has become an essential part of modern education, and e-learning websites provide learners with the opportunity to learn from anywhere, at any time.

Learning Management Systems (LMSs) serve as the foundation of many online educational solutions, providing a structured digital environment for course delivery, communication, assessment, and performance tracking. These systems are particularly valuable in addressing educational gaps in scenarios where traditional education methods may fall short.

In this project, we aim to design and develop a feature-rich, web-based Learning Management System tailored to support virtual schooling experiences. Our platform will offer tools for course management, student and instructor communication, handling assignments and exams, parental oversight, secure payments, and student placement through proficiency exams when needed, with scalability and adaptability in mind.

As a primary use case for this system, we propose using the platform to support Arabic language education for Palestinian children living abroad. These students often face challenges finding qualified Arabic instructors in non-Arabic-speaking countries. By enabling structured class enrollment, instructor-led learning, and parental monitoring, our platform can be effectively adapted to serve as an online private school for Arabic instruction.

2 PROPOSED SOLUTION

We aim to design and develop **Sconce**, a responsive **web-based Learning Management System** (LMS) that supports structured, multi-role online education. While the platform is broadly applicable to various educational contexts, it is initially designed to meet the specific needs of Arabic language instruction for Palestinian children living abroad in non-Arabic-speaking countries. Sconce provides instructors, students, parents, and administrators with the tools needed to deliver, monitor, and participate in organized learning experiences that can adapt to diverse subject areas and academic programs.

2.1 SYSTEM USERS

2.1.1 Students

Individuals who want to learn Arabic, they know Arabic of different levels.

Primary Goals:

- Learn Arabic through online classes.
- Track learning process.
- Communicate with instructors.

2.1.2 Instructors

Qualified and specialized Arabic instructors.

Primary Goals:

- Have tools to empower their teaching methods.
- Communicate with students and their parents.
- Receive payments.

2.1.3 Parents

Parents of students who are learning Arabic using the platform.

Primary Goals:

- Monitor the learning progress of their child.
- Communicate with instructors.

2.1.4 Managers

Academy Administrators, managing the platform, ensuring smooth operations, and user support.

Primary Goals:

- Oversee user activity.
- Handle problems or issues.
- Manage content, policies, and reports.
- Manage accounts (e.g. hire instructors).

2.1.5 Guests

People browsing the website without signing up yet. They can register, apply for a job as an instructor, and request information.

2.2 FUNCTIONAL REQUIREMENTS

This section outlines the functional requirements for each of the active users separately. The student, the instructor, the parent, the manager, and the guest.

2.2.1 Student Functional Requirements

Register and Set up Profile

Students can register, log in, and set up their profiles.

Take Proficiency Exam

Newly registered students should take a proficiency exam to determine their Arabic level**.**

Engage with Course

Students can attend classes, submit assignments, and take exams provided by their instructor.

View and Manage Schedule

Students can view their calendar and manage their schedule and tasks.

Receive Reminders and Notifications

Students should receive notifications for new uploaded coursework, notifications on completing coursework, and reminders for deadline approach.

Access and Track Coursework

Students can view assigned, missing, and submitted coursework.

Attend Virtual Classes

Students can attend their classes online through an integrated third-party system.

Communicate with Instructors

Students can directly message their course instructor.

Join Students Community

Students can join a community to exchange knowledge and questions.

Submit Ratings and Reviews

Students can rate and review each course (level) they complete.

Make Payments

Student can pay for their courses.

Request Drop-out

Students can request a drop-out, and their request is reviewed by the managers.

For a more detailed understanding of these functionalities, refer to the corresponding student use case diagram, fully developed use case descriptions, activity diagrams, and system sequence diagrams provided in Chapter 4: Design and Architecture.

2.2.2 Instructor Functional Requirements

Log in and Set up Profile

Instructors can log in, and set up their professional profiles.

Manage Teaching Content

Instructors should have tools for uploading material, assignments, create exams, and evaluate them.

View Student Performance

Instructors can monitor attendance and student learning progress.

Receive and Track Payments

Instructors receive their incomings and can view payment log.

Communicate with Students and Parents

Instructors can engage with students and their parents through messages.

For a more detailed understanding of these functionalities, refer to the corresponding instructor use case diagram, fully developed use case descriptions, activity diagrams, and system sequence diagrams provided in Chapter 4: Design and Architecture.

2.2.3 Parent Functional Requirements

Register and Set up Profile

Parents can register, log in, and set up their profile.

Monitor Child's Progress

Parents can monitor their child's attendance and learning progress.

Communicate with Instructors

Parents can directly message their child's instructor.

For a more detailed understanding of these functionalities, refer to the corresponding parent use case diagram, fully developed use case descriptions, activity diagrams, and system sequence diagrams provided in Chapter 4: Design and Architecture.

2.2.4 Manager Functional Requirements

Log in

Managers can log into their accounts.

Add and Remove Courses

Managers can add and remove courses.

Review and Manage Instructor Applications

Managers can review, accept, and reject instructor job applications.

Assign or Remove Instructor to/from Course

Managers can assign instructors to courses, as well as remove them.

Post Announcements and News

Managers can post announcements and news, whether publicly in the academy's main page or internally to students and instructors.

Edit Static Pages

Managers can edit the public content on the academy's main page like About and FAQ.

Update Terms of Service

Managers can update terms of service and policies.

Manage Drop-out Requests

Managers can review, accept, reject drop out requests from students.

Suspend Accounts

Managers can suspend accounts for a specified period of time.

Moderate Community

Managers can moderate the community, approve or reject posts, and manage reported content.

For a more detailed understanding of these functionalities, refer to the corresponding manager use case diagram, fully developed use case descriptions, activity diagrams, and system sequence diagrams provided in Chapter 4: Design and Architecture.

2.2.5 Guest Functional Requirements

View Public Content

Guests can view public content on the website, such as programs and courses overview, the highlighted features of the platform sections, about, and contact information.

Register as Student or Parent

Guests can register to become a student or a parent.

Apply for a Job as Instructor

Guests can fill and submit an application form to become an instructor.

Request Information

Guests can request information by submitting an inquiry form.

For a more detailed understanding of these functionalities, refer to the corresponding guest use case diagram, fully developed use case descriptions, activity diagrams, and system sequence diagrams provided in Chapter 4: Design and Architecture

2.3 NON-FUNCTIONAL REQUIREMENTS

- **Performance:** The system must perform efficiently to handle multiple users and transactions simultaneously without significant delays.
- **Scalability:** The system should maintain acceptable levels of performance as the number of users or volume of data grows.
- **Security:** The system must implement strong security measures to protect users' data privacy and transactions.
- **Maintainability:** The codebase should follow clean, modular architecture to allow for easy debugging, updating, and adding new features.
- **Availability:** The system should be available at all times with minimal downtime to ensure uninterrupted learning and teaching.
- **Usability:** The website must be intuitive and easy to navigate for all age groups of users, and support both Arabic and English interfaces.
- **Compatibility:** The system must be compatible with various devices and browsers.

2.4 FULLY DEVELOPED USE CASE DESCRIPTION

2.4.1 Student Fully Developed Use Case Descriptions

2.4.1.1 Student Register Fully Developed Use Case Description

Table 4.1: Student Register Fully Developed Use Case Description

Use Case Name	**Register as Student**
Scenario	A new student registers for an account on the platform.
Triggering Event	A user opens the registration page and submits their registration form.
Brief Description	A prospective student fills out a registration form with their name, email, and password. The system validates the input, checks for duplicate email, and then sends a verification email via a third-party email service. After the student verifies their email, they are prompted to complete a payment. Once payment is successful through a payment gateway, the system activates their account and redirects them to their dashboard.
Actors	- Primary: Student - Secondary: Email Service (third party)
Related Use Cases	- Log in - Take Proficiency Exam - Verify Email - Make Payment
Stakeholders	- Students – want a smooth registration experience. - Managers – want valid, verified users. - Email Service – responsible verifying the email.
Preconditions	- Student has not already registered (email must be unique). - Registration page is accessible. - Email service is available.
Postconditions	- Account is created and verified. - Payment is processed successfully

	- Student is redirected to their dashboard.	
Flow of Activities	Actor's Action	System's Response
	1. Student opens the registration page. 2. Student enters name, email, password. 3. Student clicks "Register". 4. Student clicks verification link. 5. Student submits payment Info. 6. Select Yes or No. 7. Student clicks "Consent and Continue". 8. Student is redirected to dashboard.	1.1 System displays form. 3.1 System validates form inputs. 3.2 System checks if email already exists. 3.3 System sends email verification request to Email Service. 4.1 System Receives a status. 4.2 System checks the status. 4.3 If valid, system updates account status to verified. 4.4 System displays payment interface. 5.1 System sends payment request to Payment Gateway. 5.2 Payment Gateway returns success/failure 5.3 If successful, system creates a student account. 5.4 Prompt: “Would you like a parent to monitor your progress?”. 6.1 If Yes, system generates a secure parent-linking code. 6.2 Display instructions for parent registration with the code. 7.1 System displays dashboard screen.

2.4.1.2 Student Log in Fully Developed Use Case Description

Table 4.2: Student Log in Fully Developed Use Case Description

Use Case Name	**Log in**	
Scenario	A registered student logs into their account to access the platform dashboard.	
Triggering Event	The student opens the login page and submits their email and password.	
Brief Description	The student navigates to the login page, enters their registered email and password, and submits the form. The system validates the inputs. If valid, the student is redirected to the dashboard. If the inputs are invalid, an error message is displayed.	
Actors	Student	
Related Use Cases	- Register - Take Proficiency Exam	
Stakeholders	- Student – wants fast access to their learning environment. - Managers – want user sessions to be properly handled.	
Preconditions	- Student has already registered. - Login page is accessible.	
Postconditions	- A valid session is initiated for the student. - Student is redirected to the dashboard.	
Flow of Activities	Actor's Action	System's Response
	1 Student opens the login page. 2 Student enters email and password. 3 Student clicks "Login". 4 Student sees dashboard screen.	1.1 System displays login form. 3.1 System validates form inputs. 3.2 If valid, generate session token. 3.3 Redirect student to dashboard.

2.4.1.3 Student Take Proficiency Exam Fully Developed Use Case Description

Table 4.3: Student Take Proficiency Exam Fully Developed Use Case Description

Use Case Name	**Take Proficiency Exam**	
Scenario	A student completes the initial placement exam to determine their appropriate level in the curriculum.	
Triggering Event	The student selects “Take Proficiency Exam” from their dashboard.	
Brief Description	The student begins the proficiency exam, completes and submits it. The system determines whether the exam should be auto-corrected or reviewed manually by an instructor. Once evaluated, the system calculates the student's level, stores it in their profile, and notifies the student of the result.	
Actors	- Primary: Student - Secondary: Instructor (if manual evaluation is needed)	
Related Use Cases	- Register - Log in	
Stakeholders	- Students – want accurate, fair placement. - Instructors – may review complex or subjective answers. - Managers – need valid level data for course placement.	
Preconditions	- Student is registered and logged in. - Student has not already taken the exam. - Exam content is available and assigned.	
Postconditions	- Student’s exam is evaluated (automatically or manually). - Assigned level is calculated and stored in their profile. - Student is notified of their placement level.	
Flow of Activities	Actor's Action	System's Response
	1 Student clicks “Take Proficiency Exam”.	1.1 System checks if exam already taken.

	2 Student answers questions. 3 Student submits exam. 4 Student sees dashboard screen.	1.2 If not, system loads exam questions. 3.1 System receives and stores answers. 3.2 System determines evaluation technique (auto/manual). 3.3 If auto: system evaluates answers. 3.4 If manual: forward exam to instructor. 3.5 Instructor evaluates answers (manual path). 3.6 System calculates final score or result. 3.7 System assigns student level. 3.8 System stores level in student profile. 3.9 System displays assigned level to student.

2.4.1.4 Student Take Exam Fully Developed Use Case Description

Table 4.4: Student Take Exam Fully Developed Use Case Description

Use Case Name	**Take Exam**
Scenario	The student takes an exam that has been assigned to them as part of a course. After completing the exam, the result is recorded and made available to the student.
Triggering Event	The student clicks “Start Exam” from the course page.
Brief Description	A student accesses and takes an exam assigned to their course. Once the student starts the exam, a timer is started. The student may either submit the exam manually or the system may auto-submit when time expires. Based on the exam type, answers are either auto-evaluated or sent to an instructor for manual grading. The system calculates the final grade and

<table>
<tr><td></td><td colspan="2">displays it to the student along with feedback (optional).</td></tr>
<tr><td>Actors</td><td colspan="2">- Primary: Student
- Secondary: Instructor (if manual evaluation is needed)</td></tr>
<tr><td>Related Use Cases</td><td colspan="2">- Engage with Course
- View Grades</td></tr>
<tr><td>Stakeholders</td><td colspan="2">- Students – want an exam interface that is accessible, easy to navigate, and submission confirmation.
- Instructor – needs tools to review, grade, and provide feedback efficiently.</td></tr>
<tr><td>Preconditions</td><td colspan="2">- Student is logged in.
- Student is enrolled in the course.
- Exam is assigned and active.</td></tr>
<tr><td>Postconditions</td><td colspan="2">- Exam is submitted (manually or automatically).
- Evaluation is completed (auto or manual).
- Grade and feedback are stored and displayed to student.
- Exam attempt is recorded.</td></tr>
<tr><td rowspan="2">Flow of Activities</td><td>Actor's Action</td><td>System's Response</td></tr>
<tr><td>1 Student opens course page.
2 Student clicks "Start Exam".
3 Student answers questions.
4 Student clicks “Submit Exam” (if before timeout).
5 Student views result and feedback.</td><td>1.1 System loads available exams.
2.1 System checks availability.
2.2 If available, loads questions and starts timer.
3.1 Timer runs in background.
3.2 If time expires first:
3.3 System auto-submits current answers.
4.1 System checks evaluation method.
4.2 If auto: evaluates answers.
4.3 If manual: forwards to instructor.
4.4 Instructor evaluates and submits grade.</td></tr>
</table>

		4.5 System stores grade and feedback. 4.6 System displays result in Grades section.

2.4.1.5 Student Submit Assignment Fully Developed Use Case Description

Table 4.5: Student Submit Assignment Fully Developed Use Case Description

Use Case Name	**Submit Assignment**
Scenario	A student submits a completed assignment for a course they are enrolled in.
Triggering Event	The student selects an assignment from the course page and clicks "Submit" after uploading their response.
Brief Description	A student accesses an available assignment and uploads their work. The system validates the file, stores the submission, and records the submission time. The system then checks if the submission was on time or late based on the assignment deadline. A confirmation is displayed, and the instructor is notified.
Actors	Student
Related Use Cases	- Engage with Course - View Grades
Stakeholders	- Student – expects a clear, reliable interface to submit work and receive confirmation. - Instructor – needs to access submissions in a timely manner and distinguish late entries.
Preconditions	- Student is logged in. - Student is enrolled in the course. - The assignment is available.
Postconditions	- The submission is saved and linked to the student. - A timestamp is recorded. - The submission is marked as either “On Time” or “Late”. - Instructor is notified.

Flow of Activities	Actor's Action	System's Response
	1 Student opens course page. 2 Student clicks on an assignment. 3 Student uploads file or enters response. 4 Student clicks "Submit".	2.1 System checks if assignment is available. 2.2 If available, system shows submission form. 4.1 System validates file type and size. 4.2 If valid, system stores submission. 4.3 System logs submission time. 4.4 System checks deadline. 4.5 Marks submission as "On Time" or "Late". 4.6 System notifies instructor. 4.7 Displays confirmation to student.

2.4.1.6 Student Join Community Fully Developed Use Case Description

Table 4.6: Student Join Community Fully Developed Use Case Description

Use Case Name	**Join Community**
Scenario	A student joins the global platform community to participate in discussions and access peer interaction features.
Triggering Event	The student clicks the "Join Community" button from the community page.
Brief Description	The student initiates a request to join the program's community. The system checks whether the student is permitted to join and whether they are already a member. If eligible and not already joined, the student is added to the community and their access permissions are initialized. A success message is displayed.

Actors	Student	
Related Use Cases	- View Community Posts. - Make a Post. - Comment on Post.	
Stakeholders	- Student – expects the ability to access community discussion features with a clear joining process. - Manager – needs control over who can access and participate in the community.	
Preconditions	- Student is logged in. - The community exists. - The student has not already joined.	
Postconditions	- Student is a member of the community. - Student's access permissions are initialized. - System confirms successful membership.	
Flow of Activities	Actor's Action	System's Response
	1 Student navigates to the community page. 2 Student clicks "Join Community".	1.1 System loads the community page. 2.1 System checks if student is permitted to join. 2.2 If not permitted, system displays denial message. 2.3 If permitted, system checks if student is already a member. 2.4 If already a member, system displays appropriate message. 2.5 If not a member, system adds student to community. 2.6 System initializes access permissions. 2.7 System displays success message.

2.4.1.7 Student Message Instructor Fully Developed Use Case Description

Table 4.7: Student Message Instructor Fully Developed Use Case Description

Use Case Name	**Message Instructor**	
Scenario	A student sends a message to the instructor of their enrolled course, optionally attaching a file for reference or clarification.	
Triggering Event	The student selects the "Message Instructor" option from the course page.	
Brief Description	A student initiates a message to their course instructor. The system checks enrollment and verifies an instructor is assigned to the course. The student composes a message, optionally adds an attachment, and submits it. The message is stored and the instructor is notified.	
Actors	- Primary Actor: Student - Secondary Actor: Instructor (receives the message)	
Related Use Cases	- Submit Assignment (shares attachment logic) - View Course	
Stakeholders	- Student – expects to communicate with the instructor in an accessible way, with supporting files. - Instructor – expects to be notified promptly of incoming student messages. - Managers – want messages to be stored and attributed correctly.	
Preconditions	- Student is logged in. - Student is enrolled in the course. - Course has an assigned instructor. - Messaging feature is enabled.	
Postconditions	- Message is stored in the system and linked to both student and instructor. - Instructor is notified.	
Flow of Activities	Actor's Action	System's Response
	1 Student opens course page.	2.1 System verifies course enrollment.

	2 Student clicks "Message Instructor". 3 Student writes a message. 4 Student optionally uploads an attachment. 5 Student clicks "Send".	2.2 System checks if instructor is assigned. 4.1 System validates attachment format and size. 5.1 System stores message content and attachments. 5.2 System notifies instructor of new message. 5.3 System displays confirmation to student.

2.4.1.8 Student Request Drop-out Fully Developed Use Case Description

Table 4.8: Student Request Drop-out Fully Developed Use Case Description

Use Case Name	**Request Drop-out**
Scenario	A student submits a request to leave the course or withdraw from the academy.
Triggering Event	The student clicks the "Request Drop-out" option from their account settings.
Brief Description	The student navigates to their account settings and submits a drop-out request form. The system checks whether there is already a pending request. If not, the student fills out the form and submits it. The request is stored with a status of "Pending" and a notification is sent to a manager for review. A confirmation is displayed to the student.
Actors	- Primary Actor: Student - Secondary Actor: Manager (receives notification for review)
Related Use Cases	- Manager: Manage Drop-out Requests
Stakeholders	- Student – expects a smooth process to request withdrawal and receive confirmation.

	- Manager – needs timely alerts to process requests and maintain student records.	
Preconditions	- Student is logged in. - Student is actively enrolled. - No other drop-out request is currently pending for this student.	
Postconditions	- A drop-out request is stored and marked “Pending”. - The manager is notified. - The student sees confirmation that the request was sent.	
Flow of Activities	Actor's Action	System's response
	1 Student logs in. 2 Student navigates to account settings. 3 Student clicks “Request Drop-out”. 4 Student fills out the request form. 5 Student clicks “Submit”.	3.1 System checks for existing pending request. 3.2 If one exists, system shows denial message. 3.3 If not, system displays form interface. 5.1 System stores request with status “Pending”. 5.2 System notifies the manager. 5.3 System shows confirmation to student.

2.4.1.9 Student Rate and Review Fully Developed Use Case Description

Table 4.9: Student Rate and Review Fully Developed Use Case Description

Use Case Name	**Rate and Review**
Scenario	A student submits a star rating and optional written feedback for a completed course.
Triggering Event	The student clicks the “Rate and Review” option on a completed course page.
Brief Description	After completing a course, the student may submit a rating and an optional comment. The system verifies

	course completion and ensures the student hasn't already submitted a review. Upon successful input, the rating and review are saved and a confirmation is displayed.	
Actors	Student	
Related Use Cases	- View Completed Courses	
Stakeholders	- Student – wants to express feedback and acknowledge good teaching. - Manager – may use ratings to assess course quality.	
Preconditions	- Student is logged in. - Student has completed the course. - No previous review for the same course has been submitted.	
Postconditions	- Rating and comment are stored and linked to student and course. - Confirmation is shown to the student.	
Flow of Activities	Actor's Action	System's Response
	1. Student navigates to completed course. 2. Student clicks "Rate and Review". 3. Student selects rating (1–5 stars). 4. Student writes optional comment. 5. Student clicks submit.	1.1 System confirms course completion. 2.1 System checks for existing review. 2.2 If review exists, system displays denial message. 2.3 If not, system displays input fields. 5.1 System stores rating and review. 5.2 System displays confirmation.

2.4.1.10 Student Create Post Fully Developed Use Case Description

Table 4.10: Student Create Post Fully Developed Use Case Description

Use Case Name	**Create Post in Community**
Scenario	Student submits a new post to the community feed.

<table>
<tr><td>Triggering Event</td><td colspan="2">Student clicks the “Create Post” button and submits the content.</td></tr>
<tr><td>Brief Description</td><td colspan="2">A logged-in student visits the community section and creates a new post by writing content and optionally attaching an image. The system validates the content, stores the post, and publishes it to the community feed if successful.</td></tr>
<tr><td>Actors</td><td colspan="2">Student</td></tr>
<tr><td>Related Use Cases</td><td colspan="2">- Join Community
- View Community Feed</td></tr>
<tr><td>Stakeholders</td><td colspan="2">- Student: Wants to express ideas, share thoughts, or ask questions to peers and instructors.
- Manager: Monitor and ensure that content is appropriate and meets platform guidelines.</td></tr>
<tr><td>Preconditions</td><td colspan="2">- Student is logged in and already a member of the community.</td></tr>
<tr><td>Postconditions</td><td colspan="2">- Post is stored in the system.
- Post is published to the community feed.</td></tr>
<tr><td rowspan="2">Flow of Activities</td><td>Actor's Action</td><td>System's Response</td></tr>
<tr><td>1. Navigate to the Community section.
2. Click “Create Post”.
3. Fill out post content, optionally attach image.
4. Click “Post”.</td><td>2.1 Display Post Box.
4.1 Check if post exceeds maximum text size.
4.2 If too long, display error.
4.3 If valid, check image format and size.
4.4 If invalid, display error.
4.5 Store post to database.
4.6 Display post on community feed.
4.7 Display confirmation: “Your post has been published”.</td></tr>
</table>

2.4.2 Instructor Fully Developed Use Case Descriptions

2.4.2.1 Instructor Log in Fully Developed Use Case Description

Table 4.11: Instructor Log in Fully Developed Use Case Description

Use Case Name	**Log in**	
Scenario	A registered instructor logs into their account to access their dashboard.	
Triggering Event	The instructor opens the login page and submits their email and password.	
Brief Description	The instructor navigates to the login page, enters their registered email and password, and submits the form. The system validates the inputs. If valid, the student is redirected to the dashboard. If the inputs are invalid, an error message is displayed.	
Actors	Instructor	
Related Use Cases	- Apply for Job	
Stakeholders	- instructor – wants fast access to their teaching environment. - Managers – want user sessions to be properly handled.	
Preconditions	- Instructor is already approved and hired. - Login page is accessible.	
Postconditions	- A valid session is initiated for the instructor. - Instructor is redirected to their dashboard.	
Flow of Activities	Actor's Action	System's Response
	1. Instructor opens the login page. 2. Instructor enters email and password. 3. Instructor clicks "Login". 4. Instructor sees dashboard screen.	1.2 System displays login form. 3.4 System validates form inputs. 3.5 If valid, generate session token. 3.6 Redirect instructor to dashboard.

2.4.2.2 Instructor Upload Course Content Fully Developed Use Case Description

Table 4.12: Instructor Upload Course Content Fully Developed Use Case Description

Use Case Name	**Upload Course Content**	
Scenario	Instructor uploads instructional material such as files, descriptions, and links to a specific course section.	
Triggering Event	Instructor selects the "Upload Content" option from the course interface.	
Brief Description	This use case allows an instructor to upload content (files, links, descriptions) to a course. The system verifies the submission, stores the content, and provides confirmation or error feedback accordingly.	
Actors	Instructor	
Related Use Cases	- Log in - Manage Course Sections	
Stakeholders	- Instructors: Need to upload and organize course material efficiently. - Students: Depend on accurate, accessible content for effective learning.	
Preconditions	- Instructor is logged in. - Instructor is authorized to upload content to the course.	
Postconditions	- Course content is uploaded and visible to students enrolled in the course.	
Flow of Activities	Actor's Action	System's Response
	1. Instructor selects "Upload Content." 2. Instructor fills out and submits form.	1.1 System displays upload form (Title, Description, Files, Links). 2.1 Receive content details and files. 2.2 Validate content fields and file formats. 2.3 If valid, store content.

		2.4 Confirm success with "Content has been uploaded" message. 2.5 Display uploaded content to instructor. 2.6 If invalid, return appropriate error message.

2.4.2.3 Instructor Create Exam Fully Developed Use Case Description

Table 4.13: Instructor Create Exam Fully Developed Use Case Description

Use Case Name	**Create Exam**
Scenario	Instructor creates and configures an exam for a course, including questions, media, and evaluation settings.
Triggering Event	Instructor selects "Create Exam" from a course page.
Brief Description	This use case describes the process of creating an exam by the instructor. It includes entering exam metadata, adding various types of questions, assigning marks, validating data, and saving the exam. After creation, the system stores the exam and notifies enrolled students.
Actors	Instructor
Related Use Cases	- Upload Assignment - Upload Material
Stakeholders	- Instructors: Need flexible exam-building tools to assess students effectively. - Students: Expect timely access to exams. - Managers: Ensure exam access aligns with academic structure.
Preconditions	- Instructor is authenticated and assigned to the course. - Course exists in the system and has enrolled students.
Postconditions	- Exam is saved to the database and linked to the course.

	- Students are notified that the exam is available.	
Flow of Activities	Actor's Action	System's Response
	1. Instructor logs in and navigates to course page. 2. Clicks “Create Exam”. 3. Enters exam title and description. 4. Selects due date and time limit. 5. Clicks “Add Question.” 6. Chooses question type. 7. Enters question text. 8. Uploads media (optional). 9. Adds answer choices. 10. Selects character to show hint (optional). 11. Writes hint text (optional). 12. Assigns marks. 13. Instructor repeats to add more questions (optional). 14. Clicks “Save Exam.”	2.1 Display exam creation interface. 4.1 Validate entered exam information. 4.2 If invalid, display error and terminate. 4.3 If valid, display question entry options. 12.1 Validate question fields. 12.2 If invalid, display error and terminate. 12.3 If valid, add question to exam draft. 14.1 Save exam and all questions to database. 14.2 Notify students about new exam. 14.3 Display confirmation message.

2.4.2.4 Instructor Message Student Fully Developed Use Case Description

Table 4.14: Instructor Message Student Fully Developed Use Case Description

Use Case Name	**Message Student**
Scenario	Instructor initiates and sends a message (with optional attachment) to a student enrolled in their course.
Triggering Event	Instructor chooses to send a message to a student from the course page.

<table>
<tr><td>Brief Description</td><td colspan="2">This use case describes the process where an instructor sends a message to a student via the messaging system. The message may include an optional attachment. The system validates authorization, confirms the student-instructor relationship, and uses Firebase Realtime Database to store and deliver the message.</td></tr>
<tr><td>Actors</td><td colspan="2">Instructor (Primary)

Firebase Database (Supporting System)</td></tr>
<tr><td>Related Use Cases</td><td colspan="2">None.</td></tr>
<tr><td>Stakeholders</td><td colspan="2">- Instructor: Wants to communicate efficiently with students in assigned courses.
- Student: Receives direct communication and clarifications.</td></tr>
<tr><td>Preconditions</td><td colspan="2">- Instructor is authenticated and logged in.
- Instructor is assigned to at least one course.
- Student is enrolled in the course.</td></tr>
<tr><td>Postconditions</td><td colspan="2">- Message is stored in Firebase Realtime Database.
- Message thread is updated (or created if new).
- Student is notified and can view the message.</td></tr>
<tr><td rowspan="2">Flow of Activities</td><td>Actor's Action</td><td>System's Response</td></tr>
<tr><td>1. Instructor logs in and opens a course page.
2. Instructor clicks “Message Student.”
3. Instructor selects a student from the course.
4. Instructor writes the message and optionally attaches a file.
5. Instructor clicks “Send.”
6. Instructor receives confirmation.</td><td>2.1 Check if instructor is assigned to the course.

2.2 If not assigned, deny access.

3.1 Check if student is enrolled.

3.2 If not enrolled, deny access.

5.1 Validate attachment (if any).

5.2 If invalid, display error.

5.3 Send message data to Firebase.</td></tr>
</table>

		5.4 Firebase stores the message and sends a webhook callback. 5.5 System verifies successful delivery and updates the thread. 5.6 Notify student of the new message.

2.4.3 Parent Fully Developed Use Case Descriptions

2.4.3.1 Parent Register Fully Developed Use Case Description

Table 4.15: Parent Register Fully Developed Use Case Description

Use Case Name	**Register as Parent**
Scenario	A parent creates an account on the platform and links it to their child using a unique linking code.
Triggering Event	A parent visits the registration page and submits the registration form with their child's linking code.
Brief Description	This use case describes how a parent can register on the platform by providing basic information (name, email, password) and a valid child linking code. The system validates the information, ensures the child isn't already linked to another parent, sends a verification email, and activates the account upon confirmation.
Actors	Parent
Related Use Cases	- Register as Student - Log in
Stakeholders	- Parents: Want to monitor their child's progress through their own account. - Students: Need to be securely linked to the correct parent.

	- Managers: Need verified and valid parent associations for communication and reporting.	
Preconditions	- The student's account must already exist. - The student's account must have a valid, unused parent linking code. - The parent must not already exist in the system using the same email.	
Postconditions	- A verified parent account is created. - The parent is linked to the corresponding student. - The parent is redirected to their dashboard after verification.	
Flow of Activities	Actor's Action	System's Response
	1. Navigate to the parent registration page. 2. Enter name, email, password. 3. Enter child's linking code. 4. Parent clicks verification link.	2.1 Validate form inputs. 2.2 Check if email already exists. 2.3 If not, prompt for parent to enter linking code. 3.1 Check if code is valid. 3.2 Check if the student is already linked to a parent. 3.3 If not, system sends email verification request to Email Service. 4.1 System Receives a status. 4.2 System checks the status. 4.3 If valid, system creates parent account. 4.4 Link parent to child. 4.5 Redirect parent to dashboard.

2.4.4 Manager Fully Developed Use Case Descriptions

2.4.4.1 Manager Add Course Fully Developed Use Case Description

Table 4.16: Manager Add Course Fully Developed Use Case Description

Use Case Name	**Add Course**	
Scenario	Manager adds a new course to the system with details such as name, capacity, schedule, and level.	
Triggering Event	Manager selects “Add Course” from the system dashboard or course management panel.	
Brief Description	This use case allows the manager to create a new course in a program by entering details like course name, student capacity, start and end dates, and the academic level. The system validates the data and either stores the course or shows an error message.	
Actors	Manager	
Related Use Cases	- Assign Instructor to Course - Edit Course Details - Remove Course	
Stakeholders	- Managers: Need to manage course availability and structure efficiently. - Students and Instructors: Depend on correct course setup for enrollment and teaching.	
Preconditions	- Manager is logged in and authorized to create courses.	
Postconditions	- A new course is added to the database and is now available for instructor assignment and student enrollment.	
Flow of Activities	Actor's Action	System's Response
	1. Manager selects “Add Course.” 2. Manager fills in the form and submits the data.	1.1 System displays a form with fields (Name, Capacity, Start Date, End Date, Level).

		2.1 System checks if the entered data is valid. 2.2 If valid, save the course data to the database. 2.3 Display confirmation message “Course has been added.” 2.4 If invalid, return error message highlighting the incorrect field(s).

2.4.4.2 Manager Assign Instructor to Course Fully Developed Use Case Description

Table 4.17: Manager Assign Instructor to Course Fully Developed Use Case Description

Use Case Name	**Assign Instructor to Course**
Scenario	A manager assigns an instructor to an existing course that currently does not have an assigned instructor.
Triggering Event	Manager selects a course and chooses to assign an instructor to it via the admin interface.
Brief Description	This use case allows the manager to assign an instructor to a course. The system verifies that the course does not already have an instructor. If the course is unassigned, the instructor is linked to it and the course information is updated.
Actors	
Related Use Cases	- Add Course - Approve Instructor Applications - Remove Instructor from Course
Stakeholders	- Managers: Ensure each course has a responsible instructor. - Instructors: Are officially assigned to relevant courses. - Students: Depend on the presence of a course instructor.
Preconditions	- Course and instructor records exist.

	- Manager is authenticated and authorized.	
Postconditions	- Course is updated with a newly assigned instructor. - Instructor gains permissions to manage course content and students.	
Flow of Activities	Actor's Action	System's Response
	1. Manager selects "Assign Instructor." 2. Manager selects a course. 3. System displays a list of available instructors. 4. Manager selects an instructor.	1.1 System displays available courses. 4.1 System checks if the course already has an instructor. 4.2 If the course has no instructor, it updates the course information. 4.3 Display success message "Instructor Assigned Successfully." 4.4 If course already has an instructor, display error message.

2.4.4.3 Manager Remove Instructor from Course Fully Developed Use Case Description

Table 4.18: Manager Remove Instructor from Course Fully Developed Use Case Description

Use Case Name	**Remove Instructor from Course**
Scenario	A manager unassigns an instructor from a course through the system's administrative interface.
Triggering Event	The manager chooses to manage a course and initiates the removal of the assigned instructor.
Brief Description	This use case allows a manager to disassociate an instructor from a course. The system updates the course record to reflect the removal and notifies the manager upon successful operation.
Actors	Manager

Related Use Cases	- Assign Instructor to Course - Add Course	
Stakeholders	- Managers: Must ensure course assignments reflect current staffing. - Instructors: Are removed from course responsibilities. - Students: May temporarily lose instructor assignment.	
Preconditions	- A valid course exists with an instructor currently assigned. - Manager is authenticated and authorized.	
Postconditions	- Instructor assignment is removed from the course record. - Course becomes available for reassignment.	
Flow of Activities	Actor's Action	System's Response
	1. Manager selects "Remove Instructor." 2. Manager selects a course. 3. Manager clicks "Remove Instructor."	1.1 System displays courses that have assigned instructors. 2.1 System displays the instructor assigned to that course. 3.1 System updates the course record, removing the instructor ID. 3.2 Display confirmation message to the manager.

2.4.4.4 Manager Post Announcement Fully Developed Use Case Description

Table 4.19: Manager Post Announcement Fully Developed Use Case Description

Use Case Name	**Post Announcement**
Scenario	A manager publishes an announcement targeted at specific course(s).

Triggering Event	The manager decides to post an announcement to notify users (e.g., students, instructors) about important information.	
Brief Description	This use case allows a manager to create and publish announcements by submitting a form with title, content, attachments, and course targeting. The system validates the data, stores the announcement, and notifies relevant users upon success.	
Actors	Manager	
Related Use Cases	- Add Course	
Stakeholders	- Managers: Need a way to distribute timely updates to users. - Students/Instructors: Depend on receiving announcements relevant to their enrolled courses.	
Preconditions	- Manager is authenticated and authorized. - One or more courses are available in the system.	
Postconditions	- The announcement is saved in the system. - Associated users are notified.	
Flow of Activities	Actor's Action	System's Response
	1. Manager navigates to "Post Announcement" page. 2. Manager fills out and submits the form.	1.1 System displays form with fields: title, content, course selection, attachments. 2.1 System validates the input. 2.2 If valid: saves announcement, sends notifications. 2.3 If invalid: displays error message. 2.4 Displays success message upon completion.

2.4.4.5 Manager Manage Job Applications Fully Developed Use Case Description

Table 4.20: Manager Manage Job Applications Fully Developed Use Case Description

Use Case Name	**Manage Job Applications**	
Scenario	Manager processes a submitted instructor application and decides whether to approve or reject it.	
Triggering Event	A manager navigates to the job applications section to review a pending application.	
Brief Description	This use case allows the manager to review instructor applications submitted by candidates. Upon review, the manager either approves the application—prompting the system to create an instructor account and notify the applicant—or rejects it, prompting a rejection email.	
Actors	Manager	
Related Use Cases	- Guest Apply for Job	
Stakeholders	- Manager: Responsible for reviewing applications and maintaining instructor quality. - Instructor Applicant: Expects timely response and onboarding if accepted.	
Preconditions	- Applicant has submitted a complete instructor application. - Manager is logged in and authorized to review applications.	
Postconditions	- If approved: Instructor account is created, and applicant is notified. - If rejected: Applicant receives a rejection notification.	
Flow of Activities	Actor's Action	System's Response
	1. Navigates to the job applications section. 2. Selects an application to review. 3. Reviews the submitted application. 4. Decides whether to approve or reject.	4.1 If rejected: Sends rejection email. 4.2 If approved: Creates instructor account. 4.3 Sends acceptance email with credentials.

2.4.5 Guest Fully Developed Use Case Descriptions

2.4.5.1 Guest Apply for Job Fully Developed Use Case Description

Table 4.21: Guest Apply for Job Fully Developed Use Case Description

Use Case Name	**Apply for Instructor Job**	
Scenario	A guest user applies to become an instructor by submitting an online application form that includes verification via email.	
Triggering Event	The guest clicks “Apply for an Instructor Job” on the website.	
Brief Description	This use case allows guest users to submit an application for an instructor position. The system validates the form, verifies the applicant’s email address, and notifies the manager once the application is successfully submitted and verified.	
Actors	- Guest - Email Service	
Related Use Cases	- Review Instructor Application	
Stakeholders	- Guest (Applicant): Expects a smooth and secure application process. - Manager: Wants to be notified when a new application is submitted. - Email Service: Delivers verification email.	
Preconditions	- Guest user is not already registered or previously submitted an application. - Instructor application form is accessible.	
Postconditions	- The application is stored in the system if the email is verified. - The manager is notified of a new instructor application.	
Flow of Activities	Actor's Action	System's Response
	1. Clicks “Apply for an Instructor Job” 2. Fills the form 3. Clicks “Submit Form”	1.1 Displays job application form 3.1 Validates form inputs

	4. Checks email and clicks verification link	3.2 Checks if email already exists 3.3 If email exists → Display error 3.4 If not, generate email verification token 3.5 Sends Email Verification Request to Email Service 3.6 Display "Please check your email to verify your account" 4.1 Receive webhook callback from Email Service 4.2 Check Status 4.3 If invalid → Display error 4.4 If valid → Store application, display confirmation 4.5 Notify manager

2.4.5.2 Guest Request Information Fully Developed Use Case Description

Table 4.22: Guest Request Information Fully Developed Use Case Description

Use Case Name	**Request Information**
Scenario	Guest user submits a form to request more details about the academy.
Triggering Event	Guest clicks "Get More Information" button after completing the form.
Brief Description	A visitor to the platform's public-facing homepage fills out a request information form, providing personal details and their question. Upon submission, the system validates the input, stores the request, and notifies the manager for follow-up.
Actors	Guest

Related Use Cases	None	
Stakeholders	- Guest: Seeks to learn more about the platform and expects timely follow-up communication from the academy. - Managers: Responsible for receiving and reviewing submitted information requests and responding with appropriate guidance or details about the academy.	
Preconditions	- The guest has access to the platform's homepage. - The "Request Information" form is available and accessible.	
Postconditions	- A new information request is stored in the system. - The manager is notified of the request for follow-up. - The guest sees a confirmation message on the screen.	
Flow of Activities	Actor's Action	System's Response
	1. Visit the website. 2. Navigate to the request information section. 3. Fill out the form. 4. Click "Get More Information".	4.1 Validate form fields (email, phone, etc.). 4.2 If valid, store the request in the database. 4.3 Notify the manager or support team. 4.4 Display a confirmation message to the user.

3 METHODOLOGY

In this chapter, we present the software development methodology we will be adopting in our project. We explain the Agile approach, its key principles, and the reasons it was chosen to guide the development of our system.

3.1 INTRODUCTION TO AGILE METHOD

For the development of our platform, we will be adopting the **Agile software development methodology**. Agile is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, customer feedback, and rapid delivery of functional components (Sommerville, 2016).

Agile divides the project into short development cycles called sprints, where specific features are planned, implemented, tested, and reviewed. This enables teams to adapt quickly to changing requirements and incorporate stakeholder input at every stage.

Agile is guided by the principles outlined in the Agile Manifesto, including:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a fixed plan

3.2 WHY AGILE WAS CHOSEN

Agile is particularly well-suited to our project for several reasons:

- **Evolving Requirements:** Our platform includes a variety of user types (students, parents, instructors, managers), and their needs may evolve as development progresses. Agile allows us to incorporate changes and feedback throughout the lifecycle.

- **Modular System Design:** Since the system consists of multiple interconnected components (e.g., proficiency exam, class enrollment, messaging, progress tracking), Agile helps us break the work into manageable iterations and build each module incrementally.

- **Early and Continuous Testing:** Agile promotes continuous integration and testing, which helps ensure that every feature (like placement logic or parental tracking) works as expected before moving on to the next.

- **Stakeholder Involvement:** Agile allows for **frequent reviews** with supervisors or future users, enabling us to validate design choices early and adjust quickly if needed.

- **Team Coordination:** With regular stand-ups and sprint reviews, Agile supports smooth communication and alignment within the development team.

- **Supports First-Time Development Teams:** As engineering students undertaking a large-scale project for the first time, using a flexible development methodology is more appropriate than adopting a rigid, plan-driven approach. Agile allows us to adapt to changes, learn from ongoing progress, and respond to challenges iteratively — making it well-suited to our experience level.

3.3 Implementation in Our Project

We will be organizing our work into 2-week sprints, each focused on delivering a set of related features. At the end of each sprint, we will be conducting:

- A sprint review to demonstrate progress.

- A retrospective to identify areas of improvement.

- Task re-prioritization for the next sprint based on current progress and challenges.

We will be using tools like **Trello** and **GitHub Projects** for task tracking and sprint planning.

4 TECHNOLOGIES AND DEVELOPMENT TOOLS

In this chapter, we will cover both technology stacks and environments we will be using throughout the project.

4.1 FRONTEND TECHNOLOGIES

The frontend of our platform will be developed using a combination of standard web technologies and modern JavaScript libraries. The main goal is to ensure a responsive, interactive, and user-friendly interface that is accessible across various devices.

4.1.1 HTML (HyperText Markup Language)

Used as the foundational structure for all web pages. It defines the semantic layout and content hierarchy of the user interface.

4.1.2 CSS (Cascading Style Sheets)

Responsible for styling and layout. CSS is used to define the appearance of elements, manage layout responsiveness, and ensure consistency across the platform.

4.1.3 JavaScript

Acts as the core scripting language for client-side interactivity. JavaScript enables dynamic behavior on the platform, including form validation, content updates, and UI responsiveness without reloading pages.

4.1.4 Bootstrap

A widely-used front-end library that provides pre-designed UI components and a responsive grid system. It helps accelerate development and maintain visual consistency across the application.

4.1.5 React.js

A modern JavaScript library used to build reusable and efficient user interface components. React enables the development of a single-page application (SPA) architecture, improving performance and user experience through seamless transitions and real-time data updates.

4.1.6 FullCalendar

A robust JavaScript calendar library used in the frontend to display scheduled classes, assignment deadlines, and virtual sessions. It provides interactive views (monthly, weekly, daily) and supports event filtering and integration with our internal scheduling system.

This combination allows us to create a modular and maintainable front-end codebase, while ensuring high usability and visual responsiveness for users across roles such as students, parents, instructors, and managers.

4.2 BACKEND TECHNOLOGIES

The backend of the platform is developed using ASP.NET Core, a modern, open-source web framework developed by Microsoft. It is used in combination with C# for building scalable, high-performance web APIs that handle logic and communication with the database.

4.2.1 ASP.NET Core

Serves as the main framework for developing the backend. It supports RESTful API development and is known for its performance, security, and cross-platform capabilities.

4.2.2 C#

The primary programming language used to implement backend logic, including user management, course control, messaging, and payment integration.

4.2.3 Entity Framework Core (EF Core)

An object-relational mapping (ORM) tool that allows interaction with the database using C# objects instead of SQL queries. EF simplifies data access and improves maintainability.

4.2.4 LINQ (Language Integrated Query)

Used alongside EF Core to perform queries against in-memory collections or the database in a readable and strongly-typed manner.

4.2.5 Web APIs

The backend exposes endpoints that allow the frontend (React) to interact with the server, retrieve data, and perform user actions (e.g., login, enroll in course, send messages).

This combination provides a structured, testable, and efficient backend suitable for supporting multi-role functionality and future scalability.

4.3 DATABASE TECHNOLOGY

We will be using Microsoft SQL Server as the primary relational database management system (RDBMS) for our platform. It is tightly integrated with the .NET ecosystem, ensuring smooth compatibility with EF Core and ASP.NET.

4.3.1 Microsoft SQL Server

SQL Server offers:

- Strong support for complex queries and transactions.
- Data integrity enforcement through relationships and constraints.
- High performance and scalability for structured data.
- Integrated tools for database management and security.

All core entities in the system — such as users, courses, enrollments, messages, exams, and submissions — are modeled using relational schemas in SQL Server. To check our initial database design, see Figure 4.43: Entity Relationship Diagram and Figure 4.44: Domain Model Class Diagram on pages 91 and 92.

4.4 DEVELOPMENT ENVIRONMENTS

Our team will be using the following tools and environments to build and maintain the system:

4.4.1 Visual Studio (VS)

The primary IDE for backend development, providing powerful support for C#, ASP.NET, and Entity Framework.

4.4.2 Visual Studio Code (VS Code)

Used mainly for frontend development with React and JavaScript, offering flexibility and a lightweight editing experience.

4.4.3 Git & GitHub

For version control and collaborative development. GitHub hosts the project repositories, manages branches, and facilitates issue tracking and code reviews.

These tools together provide a streamlined and productive environment for full-stack development and team collaboration.

4.5 APIS AND THIRD-PARTY SERVICES

To extend the core functionality of our platform, we will be integrating with several third-party services and APIs. These integrations will allow us to implement essential features such as email verification, secure payments, real-time messaging, and virtual class hosting without

the overhead of building such systems from scratch. The selected services are widely used, well-documented, and offer developer-friendly integration with our .NET backend and React frontend.

4.5.1 Email Verification and Authentication — SendGrid

To ensure secure and valid user registrations, we will be using **SendGrid** for email verification. SendGrid allows us to send verification emails with unique tokens to users upon registration. It supports dynamic templates, reliable delivery, and easy integration with ASP.NET Core via official SDKs. This helps us confirm user identities before activating accounts, reducing spam and fraudulent registrations.

4.5.2 Payment Gateway — Tap Payments

For handling online payments securely and reliably, we will integrate **Tap Payments**, a leading payment gateway in the MENA region. Tap provides a developer-friendly platform that supports one-time payments, multiple currencies, and secure checkout experiences. It is PCI-DSS compliant and offers fraud prevention measures to ensure transaction safety.

Tap will integrate seamlessly with our backend using RESTful APIs and webhooks. This enables our system to initiate payment sessions, track payment statuses, and handle confirmations programmatically. The platform also supports hosted checkout pages and can be customized to fit our design needs.

By using Tap Payments, we ensure a smooth and localized experience for students during course enrollment, while enabling reliable pay out capabilities for instructors where supported.

4.5.3 Messaging Service — Firebase Realtime Database

To facilitate real-time messaging between students, parents, and instructors, we will be using **Firebase Realtime Database**. Firebase enables the instant sending and receiving of messages without refreshing the page, which improves responsiveness and user experience. Its event-driven architecture allows us to create lightweight chat interfaces while maintaining real-time synchronization across devices.

4.5.4 Virtual Class Integration — Zoom

For live, instructor-led classes, we will be integrating **Zoom** to host virtual sessions. Zoom is a reliable and widely recognized video conferencing tool that supports screen sharing, chat, breakout rooms, and interactive participation. By generating meeting links through Zoom's API and associating them with scheduled classes in our platform, we provide a seamless experience for both instructors and students. With a business plan, Zoom also allows access to detailed participant reports, including attendance and session duration, which we may use for student performance analysis.

These third-party services will help us ensure scalability, security, and a professional user experience, while allowing us to focus development efforts on core learning features and user-specific flows.

See Table 6.1 in the next page for a summary that captures all technologies and tools we plan to use.

Table 6.1: Project Technologies and Development Tools

Category	Technology/Tool	Purpose
Frontend	HTML, CSS, JavaScript	Core web technologies for structure, styling, and interactivity
	Bootstrap	UI component framework for responsive and consistent design
	React.js	Frontend library for building reusable components and managing UI state
	FullCalendar	Interactive calendar library for displaying class schedules and deadlines
Backend	ASP.NET Core	Web framework for building APIs and server-side logic
	C#	Programming language for backend development
	Entity Framework Core (EF)	ORM for database interactions
	LINQ	Query syntax used with EF for database queries
	Web API	RESTful endpoints to handle frontend-backend communication
Database	Microsoft SQL Server	Relational database to store structured data like users, classes, messages
Development Environment	Visual Studio	Primary IDE for backend development
	Visual Studio Code	Lightweight editor for frontend development
	Git & GitHub	Version control and collaboration platform
APIs & Third-Party Services	SendGrid	Email verification and transactional emails
	Stripe	Payment gateway for secure tuition processing
	Firebase Realtime Database	Real-time messaging between users
	Zoom	Virtual class hosting and video conferencing

5 PROJECT TIMELINE

This chapter outlines the planned phases and key milestones of our project using an Agile-based approach. The work is organized into iterative sprints with continuous development, testing, and feedback loops. Table 7.1 and Figure 7.1 show the estimated timeline and task distribution.

Table 7.1: Project Timeline

Phase	Description	Estimated Duration
1. Extra Requirement Analysis	Gather more information from stakeholders using questionnaires and interviews.	15 Jun. – 1 Jul.
2. Possible Refinement on System Design	Modify system design and architecture (if needed).	20 Jun. – 1 Jul.
3. UI/UX Design	Create wireframes and user flows; plan role-based layouts.	1 Jul. – 1 Aug.
4. Frontend Development	Implement core frontend components using React, Bootstrap, etc.	1 Aug. – 1 Dec.
5. Backend Development	Build APIs with ASP.NET Core, implement database schema with SQL Server, integrate authentication.	15 Jul. – 1 Dec.
6. Third-Party Integration	Integrate SendGrid, Stripe, Firebase, and Zoom.	25 Jul. – 1 Dec.
7. Testing & Bug Fixing	Conduct unit tests, integration tests, UI testing, fix reported issues.	1 Dec. – 1 Jan.
8. Final Report Writing	Add acknowledgment, dedication, results, conclusion, etc.	1 Dec. – 1 Jan.
9. Proofreading	Make sure there are no errors in the document.	1 Jan. – 5 Jan.

10. Prepare Presentation & Rehearse	Prepare a well-structured presentation and rehearse for the project defend.	5 Jan. – 20 Jan.
11. Project Defend	Defend the project.	Last days of Jan.