RadiantIQ

UNIVERSITÀ
DI TRENTO
Department of
Information Engineering and Computer Science

Project's acronym:	RADIANTIQ
Project's title:	RadiantIQ
Start date:	08/03/2024

D2 - RadiantIQ Agile Methodology

WP1:	Software Specification
Task 2.1:	Define Agile Methodology
Submission date:	27/05/2024
Responsible:	RadiantIQ develop team
Version:	2.0
Status:	Completed
Author(s):	Lorenzo Cattai, Gabriele Pernici, Anh Tu Duong, Lorenzo Negut
Deliverable type:	DOCUMENT

Contents

1 Introduction

This document outlines the agile management practices for RadiantIQ project, focusing on flexibility, iterative progress, and collaboration. The document covers key components: architecture description, product backlog, testing, Git strategy, Definition of Done, and the initial sprint. The intended audiences for this document include:

- Project Managers: To track agile processes and ensure alignment with project goals.
- Developers: To understand architecture, coding standards, testing, and version control practices.
- Stakeholders: To stay informed about the project methodology for transparency and communication.

2 Architecture description

2.1 Overview

RadiantIQ is designed using a microservices architecture to ensure scalability, flexibility, and maintainability. This section provides a detailed description of the architecture, including the components, technologies used, and the communication flow between services.

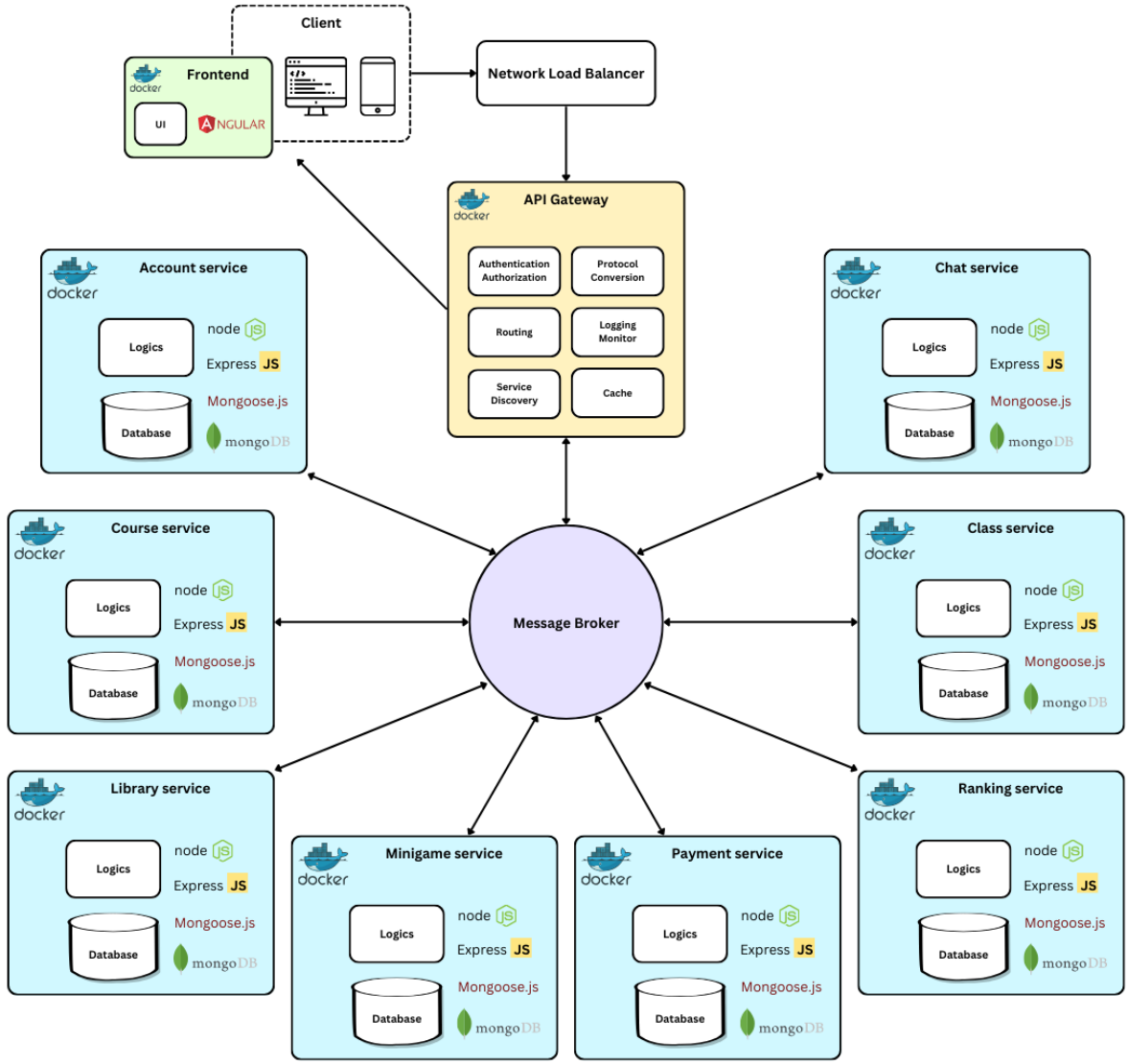

Figure 2: RadiantIQ - Architecture description diagram

2.2 Architectural Components

2.2.1 Frontend

- **Technology Stack**: Angular, NgBootstrap
- **Description**: The frontend is responsible for delivering a responsive and interactive user interface. It communicates with the backend services through the API Gateway.

2.2.2 API Gateway

Responsibilities:

- **Authentication and Authorization**: Verifies user credentials and permissions.
- **Routing**: Forwards requests to the appropriate backend services.
- **Service Discovery**: Determines the location of microservices.
- **Protocol Conversion**: Converts protocols (e.g., HTTP to WebSockets).
- **Logging and Monitoring**: Tracks and logs requests for monitoring and debugging.
- **Caching**: Stores frequently accessed data to improve performance.

2.2.3 Microservices

Each microservice is designed to handle a specific domain within the RadiantIQ platform. They are developed using Node.js and Express, and use MongoDB for data storage. Each service is deployed in a separate container, ensuring isolation and scalability.

- **Account Service**: Manages user accounts, including registration, login, and profile management.
- **Chat Service**: Handles real-time messaging between users.
- **Class Service**: Manages virtual classrooms, including scheduling and attendance.
- **Course Service**: Manages course content, enrollment, and progress tracking.
- **Library Service**: Provides access to educational resources, materials and allows users to create and share collections.
- **Minigame Service**: Create an environments for external developers to create minigames.
- **Payment Service**: Manages payment processing for course enrollments and other transactions.
- **Ranking Service**: Tracks and displays user rankings and achievements.

2.2.4 Message Broker

Description: Facilitates communication between microservices using a publish-subscribe model. Ensures that messages are reliably delivered to the appropriate services.

2.3 Communication Flow

1. **Client Request**:
   - The client (frontend) sends a request to the Network Load Balancer.
   - The Network Load Balancer forwards the request to the API Gateway.
2. **API Gateway Processing**:
   - The API Gateway authenticates the request and checks user authorization.
   - It routes the request to the appropriate service based on the endpoint and request data.
3. **Service Interaction via Message Broker**:
   - The API Gateway forwards the request to the Message Broker.
   - The Message Broker directs the request to the corresponding microservice (e.g., Course Service for course-related operations).

4. **Inter-service Communication**:
    - Services communicate with each other through the Message Broker using GraphQL API. This ensures decoupled and asynchronous communication, enhancing scalability and reliability.
5. **Response Handling**:
    - The target microservice processes the request and sends the response back to the API Gateway through the Message Broker.
    - The API Gateway returns the response to the client.

2.4 Deployment and Scalability

- Each microservice is containerized using Docker, ensuring consistent environments across development, testing, and production.
- Services can be independently scaled based on load and performance requirements.
- Kubernetes (or another orchestration tool) can be used to manage container deployment, scaling, and load balancing.
- Finally the services are deployed on a cloud provider Render.com to expose to the internet.

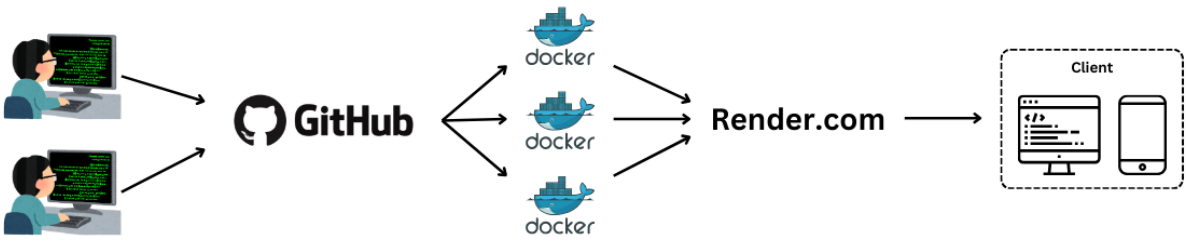

Figure 3: RadiantIQ - Develop and deploy workflow

2.5 Monitoring and Maintenance

- **Logging and Monitoring**: Integrated within the API Gateway and microservices to track performance, errors, and usage patterns.
- **Continuous Integration/Continuous Deployment (CI/CD)**: Using Github and Github Action to automate pipelines for testing and deploying new code changes to ensure rapid and reliable updates.

3 API Definition

Creating a well-defined API is crucial for ensuring clear communication between different services. This section will be a guideline for establishing a structured and consistent approach to defining APIs in RadiantIQ project.

3.1 General Principles

- **Consistency:** Ensure that naming conventions, structures, and responses are consistent across all endpoints.
- **Simplicity:** Design the API to be simple and intuitive, making it easy for everyone to understand and use.
- **GraphQL Principles:** Follow GraphQL design principles, including declarative data fetching and single endpoint structure.
- **Documentation:** Document the API comprehensively, including all queries, mutations, parameters, request and response formats, and error codes.
- **Versioning:** Include versioning in the API to allow for backward compatibility and future updates.
- **Security:** Implement appropriate security measures, such as authentication and authorization, to protect the API from unauthorized access.
- **Error Handling:** Define clear error messages and codes to help developers troubleshoot issues.
- **Testing:** Test the API thoroughly to ensure that it functions as expected and returns the correct responses.

3.2 API Schema

- **Type Definitions:** Use type definitions to represent entities in the system, following a hierarchical structure.
- **Queries and Mutations:** Use queries for fetching data and mutations for modifying data.
- **Response Codes:** Although GraphQL doesn't use HTTP status codes in the same way as REST, ensure meaningful error messages are provided in the response.
- **Request and Response Formats:** Use JSON as the default format for both request and response bodies.
- **Filtering and Sorting:** Allow users to filter and sort results based on specific criteria within the query parameters.

3.3 Authentication and Authorization

- **Authentication:** Use token-based authentication (JWT) to authenticate users and secure API endpoints.
- **Authorization:** Implement role-based access control to restrict access to certain queries and mutations based on user roles.
- **Rate Limiting:** Implement rate limiting to prevent abuse and protect the API from excessive requests.

3.4 APIs in GraphQL

3.4.1 Account

```
"""
Represents a user in the system.
"""
type User {
        """
        A unique identifier for the user.
        """
        id: ID!

        """
        The username of the user.
        """
        username: String!

        """
        The roles assigned to the user.
        """
        roles: [UserRole!]!

        """
        The first name of the user.
        """
        name: String!

        """
        The last name of the user.
        """
        surname: String!

        """
        The age of the user.
        """
        age: Int!

        """
        The date of birth of the user (optional).
        """
        dateOfBirth: Int

        """
        The gender of the user (optional).
        """
        gender: String

        """
        The recovery method for the user's account.
        """
        recoveryMethod: String!
}

"""
Enumeration of possible user roles.
"""
```

```
enum UserRole {
        """
        Student role.
        """
        STUDENT

        """
        Professor role.
        """
        PROFESSOR

        """
        Publisher role.
        """
        PUBLISHER

        """
        Developer role.
        """
        DEVELOPER

        """
        AI Supervisor role.
        """
        AI_SUPERVISOR

        """
        Tech Support role.
        """
        TECH_SUPPORT

        """
        Course Supervisor role.
        """
        COURSE_SUPERVISOR

        """
        System Admin role.
        """
        SYSTEM_ADMIN
}

"""
Represents the payload returned after authentication.
"""
type AuthPayload {
        """
        The authentication token.
        """
        token: String

        """
        The user object.
        """
        user: User
}
```

```
"""
Root query type.
"""
type Query {
        """
        Fetches the profile of the authenticated user.
        """
        userProfile: User @authenticated
}

"""
Root mutation type.
"""
type Mutation {
        """
        Logs a user into the system.
        """
        login(username: String!, password: String!): AuthPayload

        """
        Logs out the authenticated user.
        """
        logout: Boolean @authenticated

        """
        Initiates a process to recover forgotten credentials.
        """
        recoverCredentials: Boolean

        """
        Registers a new user account.
        """
        registerAccount(username: String!, password: String!,
        roles: [UserRole!]!): User

        """
        Deletes the authenticated user's account.
        """
        deleteAccount(password: String!): Boolean @authenticated

        """
        Modifies core settings of the system.
        """
        modifyCoreSettings(password: String!,
        newSettings: [String!]!): Boolean @authenticated

        """
        Modifies secondary settings of the system.
        """
        modifySecondarySettings(newSettings: [String!]!):
        Boolean @authenticated

        """
        Modifies AI theming settings.
        """
```

```
        modifyAITheming(prompt: String!): Boolean @authenticated @role(
        roles: [STUDENT, AI_SUPERVISOR, TECH_SUPPORT])

        """
        Adds new roles to the user.
        """
        addUserRoles(newRoles: [UserRole!]!): Boolean @authenticated

        """
        Removes roles from the user.
        """
        removeUserRoles(oldRoles: [UserRole!]!): Boolean @authenticated
}

"""
Directive to restrict access to authenticated users.
"""
directive @authenticated on FIELD_DEFINITION

"""
Directive to restrict access based on user roles.
"""
directive @role(
"""
The roles allowed to access the field or object.
"""
roles: [UserRole!]!
) on FIELD_DEFINITION | OBJECT
```

3.4.2 Course

```
schema {
        query: Query
        mutation: Mutation
}

"""
Root Query for fetching course data.
"""
type Query {
        """
        View a specific course by its ID.
        """
        viewCourse(id: ID!): Course @role(roles: [STUDENT, PROFESSOR,
        PUBLISHER, DEVELOPER, AI_SUPERVISOR, TECH_SUPPORT,
        COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        List all available courses.
        """
        listCourses: [Course] @role(roles: [STUDENT, PROFESSOR,
        PUBLISHER, DEVELOPER, AI_SUPERVISOR, TECH_SUPPORT,
        COURSE_SUPERVISOR, SYSTEM_ADMIN])
}

"""
```

```
    Root Mutation for modifying course data.
    """
    type Mutation {
            """
            Create a new course.
            """
            createCourse(input: CreateCourseInput!): CourseResponse @role(
            roles: [COURSE_SUPERVISOR, SYSTEM_ADMIN])

            """
            Modify an existing course.
            """
            modifyCourse(input: ModifyCourseInput!): CourseResponse @role(
            roles: [COURSE_SUPERVISOR, SYSTEM_ADMIN])

            """
            Delete a course by its ID.
            """
            deleteCourse(id: ID!): DeleteCourseResponse @role(roles:
            [COURSE_SUPERVISOR, SYSTEM_ADMIN])

            """
            Archive a course by its ID.
            """
            archiveCourse(id: ID!): ArchiveCourseResponse @role(roles:
            [COURSE_SUPERVISOR, SYSTEM_ADMIN])

            """
            Add a review to a course.
            """
            reviewCourse(input: ReviewCourseInput!): ReviewResponse @role(
            roles: [STUDENT, PROFESSOR])

            """
            Remove a review from a course.
            """
            removeReview(courseId: ID!, reviewId: ID!): RemoveReviewResponse
            @role(roles: [SYSTEM_ADMIN])
    }

    """
    Input type for creating a new course.
    """
    input CreateCourseInput {
            title: String!
            description: String!
            content: String!
            authorId: ID!
            isDraft: Boolean
    }

    """
    Input type for modifying an existing course.
    """
    input ModifyCourseInput {
            id: ID!
```

```
        title: String
        description: String
        content: String
        isDraft: Boolean
}

"""
Response type for course-related mutations.
"""
type CourseResponse {
        success: Boolean!
        message: String
        course: Course
}

"""
Response type for deleting a course.
"""
type DeleteCourseResponse {
        success: Boolean!
        message: String
}

"""
Response type for archiving a course.
"""
type ArchiveCourseResponse {
        success: Boolean!
        message: String
}

"""
Input type for adding a review to a course.
"""
input ReviewCourseInput {
        courseId: ID!
        review: String!
        reviewerId: ID!
}

"""
Response type for review-related mutations.
"""
type ReviewResponse {
        success: Boolean!
        message: String
        review: Review
}

"""
Response type for removing a review.
"""
type RemoveReviewResponse {
        success: Boolean!
        message: String
}
```

```
"""
Representation of a course.
"""
type Course {
        id: ID!
        title: String!
        description: String!
        content: String!
        author: User!
        isDraft: Boolean!
        isArchived: Boolean!
        createdAt: String!
        updatedAt: String!
        reviews: [Review!]
        globalRanking: Float
}

"""
Representation of a review.
"""
type Review {
        id: ID!
        content: String!
        reviewer: User!
        createdAt: String!
}
```

3.4.3 Class

```
schema {
        query: Query
        mutation: Mutation
}

"""
Root Query for fetching class data.
"""
type Query {
        """
        View a specific class by its ID.
        """
        viewClass(id: ID!): Class @role(roles:
        [STUDENT, PROFESSOR, PUBLISHER, DEVELOPER, AI_SUPERVISOR,
        TECH_SUPPORT, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        List all available classes.
        """
        listClasses: [Class] @role(roles:
        [STUDENT, PROFESSOR, PUBLISHER, DEVELOPER, AI_SUPERVISOR,
        TECH_SUPPORT, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        Display attendees of a specific class.
```

```
        """
        displayClassAttendees(classId: ID!): [User] @role(roles:
        [STUDENT, PROFESSOR])

        """
        Display performance statistics for a specific class.
        """
        displayClassStatistics(classId: ID!): ClassStatistics @role(roles:
        [PROFESSOR])
}

"""
Root Mutation for modifying class data.
"""
type Mutation {
        """
        Create a new class.
        """
        createClass(input: CreateClassInput!): ClassResponse @role(roles:
        [PROFESSOR])

        """
        Modify an existing class.
        """
        modifyClass(input: ModifyClassInput!): ClassResponse @role(roles:
        [PROFESSOR, SYSTEM_ADMIN])

        """
        Terminate a class by its ID.
        """
        terminateClass(id: ID!): TerminateClassResponse @role(roles:
        [PROFESSOR, SYSTEM_ADMIN])

        """
        Archive a class by its ID.
        """
        archiveClass(id: ID!): ArchiveClassResponse @role(roles:
        [PROFESSOR, SYSTEM_ADMIN])

        """
        Join a class.
        """
        joinClass(classId: ID!, studentId: ID!): JoinClassResponse @role(
        roles: [STUDENT])

        """
        Leave a class.
        """
        leaveClass(classId: ID!, studentId: ID!): LeaveClassResponse @role(
        roles: [STUDENT])
}

"""
Input type for creating a new class.
"""
input CreateClassInput {
```

```
        title: String!
        description: String!
        professorId: ID!
        studentIds: [ID!]
}

"""
Input type for modifying an existing class.
"""
input ModifyClassInput {
        id: ID!
        title: String
        description: String
        studentIds: [ID!]
}

"""
Response type for class-related mutations.
"""
type ClassResponse {
        success: Boolean!
        message: String
        class: Class
}

"""
Response type for terminating a class.
"""
type TerminateClassResponse {
        success: Boolean!
        message: String
}

"""
Response type for archiving a class.
"""
type ArchiveClassResponse {
        success: Boolean!
        message: String
}

"""
Response type for joining a class.
"""
type JoinClassResponse {
        success: Boolean!
        message: String
}

"""
Response type for leaving a class.
"""
type LeaveClassResponse {
        success: Boolean!
        message: String
}
```

```
"""
Representation of a class.
"""
type Class {
        id: ID!
        title: String!
        description: String!
        professor: User!
        students: [User!]
        isArchived: Boolean!
        createdAt: String!
        updatedAt: String!
}

"""
Statistics for a class.
"""
type ClassStatistics {
        classId: ID!
        averageScore: Float
        numberOfStudents: Int
}
```

3.4.4 Article

```
schema {
        query: Query
        mutation: Mutation
}

"""
Root Query for fetching article data.
"""
type Query {
        """
        View a specific article by its ID.
        """
        viewArticle(id: ID!): Article @role(roles:
        [STUDENT, PROFESSOR, PUBLISHER, DEVELOPER, AI_SUPERVISOR,
        TECH_SUPPORT, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        List all available articles.
        """
        listArticles: [Article] @role(roles:
        [STUDENT, PROFESSOR, PUBLISHER, DEVELOPER, AI_SUPERVISOR,
        TECH_SUPPORT, COURSE_SUPERVISOR, SYSTEM_ADMIN])
}

"""
Root Mutation for modifying article data.
"""
type Mutation {
        """
```

```
        """
        Create a new article.
        """
        createArticle(input: CreateArticleInput!): ArticleResponse @role(
        roles: [PUBLISHER])

        """
        Modify an existing article.
        """
        modifyArticle(input: ModifyArticleInput!): ArticleResponse @role(
        roles: [PUBLISHER, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        Delete an article by its ID.
        """
        deleteArticle(id: ID!): DeleteArticleResponse @role(roles:
        [PUBLISHER, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        Archive an article by its ID.
        """
        archiveArticle(id: ID!): ArchiveArticleResponse @role(roles:
        [PUBLISHER, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        Add a review to an article.
        """
        reviewArticle(input: ReviewArticleInput!): ReviewResponse @role(
        roles: [STUDENT, PROFESSOR])

        """
        Remove a review from an article.
        """
        removeReview(articleId: ID!, reviewId: ID!): RemoveReviewResponse
        @role(roles: [SYSTEM_ADMIN])
}

"""
Input type for creating a new article.
"""
input CreateArticleInput {
        title: String!
        content: String!
        authorId: ID!
}

"""
Input type for modifying an existing article.
"""
input ModifyArticleInput {
        id: ID!
        title: String
        content: String
}

"""
Response type for article-related mutations.
```

```
"""
type ArticleResponse {
        success: Boolean!
        message: String
        article: Article
}

"""
Response type for deleting an article.
"""
type DeleteArticleResponse {
        success: Boolean!
        message: String
}

"""
Response type for archiving an article.
"""
type ArchiveArticleResponse {
        success: Boolean!
        message: String
}

"""
Input type for adding a review to an article.
"""
input ReviewArticleInput {
        articleId: ID!
        review: String!
        reviewerId: ID!
}

"""
Response type for review-related mutations.
"""
type ReviewResponse {
        success: Boolean!
        message: String
        review: Review
}

"""
Response type for removing a review.
"""
type RemoveReviewResponse {
        success: Boolean!
        message: String
}

"""
Representation of an article.
"""
type Article {
        id: ID!
        title: String!
        content: String!
```

```
        author: User!
        createdAt: String!
        updatedAt: String!
        reviews: [Review!]
}

"""
Representation of a review.
"""
type Review {
        id: ID!
        content: String!
        reviewer: User!
        createdAt: String!
}
```

3.4.5 Payment

```
schema {
        mutation: Mutation
}

"""
Root Mutation for handling payments.
"""
type Mutation {
        """
        Pay a developer for their work.
        """
        payDeveloper(input: PayDeveloperInput!): PayDeveloperResponse
        @role(roles: [SYSTEM_ADMIN])
}

"""
Input type for paying a developer.
"""
input PayDeveloperInput {
        developerId: ID!
        amount: Float!
        paymentMethod: String!
}

"""
Response type for payment-related mutations.
"""
type PayDeveloperResponse {
        success: Boolean!
        message: String
}
```

3.4.6 Library

```
schema {
        query: Query
}
```

```
"""
Root Query for searching library materials.
"""
type Query {
        """
        Search for materials in the library.
        """
        searchMaterial(input: SearchMaterialInput!): [SearchResult]
        @role(roles: [STUDENT, PROFESSOR, PUBLISHER, DEVELOPER,
        AI_SUPERVISOR, TECH_SUPPORT, COURSE_SUPERVISOR, SYSTEM_ADMIN])
}

"""
Input type for searching library materials.
"""
input SearchMaterialInput {
        query: String!
        type: String
}

"""
Representation of a search result.
"""
type SearchResult {
        id: ID!
        title: String!
        type: String!
        description: String
}
```

3.4.7 Minigame

```
schema {
        query: Query
        mutation: Mutation
}

"""
Root Query for fetching minigame data.
"""
type Query {
        """
        View a specific minigame by its ID.
        """
        viewMinigame(id: ID!): Minigame @role(roles: [STUDENT, PROFESSOR,
        DEVELOPER, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        List all available minigames.
        """
        listMinigames: [Minigame] @role(roles: [STUDENT, PROFESSOR,
        DEVELOPER, COURSE_SUPERVISOR, SYSTEM_ADMIN])
}

"""
Root Mutation for modifying minigame data.
```

```
"""
type Mutation {
        """
        Create a new minigame.
        """
        createMinigame(input: CreateMinigameInput!): MinigameResponse
        @role(roles: [DEVELOPER])

        """
        Modify an existing minigame.
        """
        modifyMinigame(input: ModifyMinigameInput!): MinigameResponse
        @role(roles: [DEVELOPER])

        """
        Delete a minigame by its ID.
        """
        deleteMinigame(id: ID!): DeleteMinigameResponse @role(roles:
        [DEVELOPER, SYSTEM_ADMIN])

        """
        Archive a minigame by its ID.
        """
        archiveMinigame(id: ID!): ArchiveMinigameResponse @role(roles:
        [DEVELOPER, SYSTEM_ADMIN])

        """
        Register a minigame for a course by a developer.
        """
        registerMinigameFromDeveloper(input: RegisterMinigameFromDeveloperInput!):
        RegisterMinigameResponse @role(roles: [DEVELOPER])

        """
        Register a minigame for a course by an observer.
        """
        registerMinigameFromObserver(input: RegisterMinigameFromObserverInput!):
        RegisterMinigameResponse @role(roles: [PROFESSOR, COURSE_SUPERVISOR])

        """
        Play a minigame.
        """
        playMinigame(id: ID!): PlayMinigameResponse @role(roles: [STUDENT,
        PROFESSOR])
}

"""
Input type for creating a new minigame.
"""
input CreateMinigameInput {
        title: String!
        content: String!
        developerId: ID!
}

"""
Input type for modifying an existing minigame.
```

```
"""
input ModifyMinigameInput {
        id: ID!
        title: String
        content: String
}

"""
Response type for minigame-related mutations.
"""
type MinigameResponse {
        success: Boolean!
        message: String
        minigame: Minigame
}

"""
Response type for deleting a minigame.
"""
type DeleteMinigameResponse {
        success: Boolean!
        message: String
}

"""
Response type for archiving a minigame.
"""
type ArchiveMinigameResponse {
        success: Boolean!
        message: String
}

"""
Input type for registering a minigame by a developer.
"""
input RegisterMinigameFromDeveloperInput {
        minigameId: ID!
        courseId: ID!
}

"""
Input type for registering a minigame by an observer.
"""
input RegisterMinigameFromObserverInput {
        minigameId: ID!
        courseId: ID!
}

"""
Response type for registering a minigame.
"""
type RegisterMinigameResponse {
        success: Boolean!
        message: String
}
```

```
"""
Response type for playing a minigame.
"""
type PlayMinigameResponse {
        success: Boolean!
        message: String
}

"""
Representation of a minigame.
"""
type Minigame {
        id: ID!
        title: String!
        content: String!
        developer: User!
        createdAt: String!
        updatedAt: String!
}
```

3.4.8 Ranking

```
schema {
        query: Query
        mutation: Mutation
}

"""
Root Query for fetching ranking data.
"""
type Query {
        """
        Get a specific ranking by its ID.
        """
        getRanking(id: ID!): Ranking @role(roles:
        [STUDENT, PROFESSOR, COURSE_SUPERVISOR, SYSTEM_ADMIN])

        """
        List all rankings.
        """
        getRankings: [Ranking] @role(roles:
        [STUDENT, PROFESSOR, COURSE_SUPERVISOR, SYSTEM_ADMIN])
}

"""
Root Mutation for modifying ranking data.
"""
type Mutation {
        """
        Add a new ranking.
        """
        addRanking(input: AddRankingInput!): RankingResponse @role(roles:
        [SYSTEM_ADMIN])

        """
        Update an existing ranking.
```

```
        """
        updateRanking(input: UpdateRankingInput!): RankingResponse
        @role(roles: [SYSTEM_ADMIN])

        """
        Delete a ranking by its ID.
        """
        deleteRanking(id: ID!): DeleteRankingResponse @role(roles:
        [SYSTEM_ADMIN])
}

"""
Input type for adding a new ranking.
"""
input AddRankingInput {
        userId: ID!
        courseId: ID!
        score: Float!
}

"""
Input type for updating an existing ranking.
"""
input UpdateRankingInput {
        id: ID!
        score: Float!
}

"""
Response type for ranking-related mutations.
"""
type RankingResponse {
        success: Boolean!
        message: String
        ranking: Ranking
}

"""
Response type for deleting a ranking.
"""
type DeleteRankingResponse {
        success: Boolean!
        message: String
}

"""
Representation of a ranking.
"""
type Ranking {
        id: ID!
        user: User!
        course: Course!
        score: Float!
        createdAt: String!
        updatedAt: String!
}
```

3.4.9 Chat

```
schema {
        mutation: Mutation
}

"""
Root Mutation for handling chat interactions.
"""
type Mutation {
        """
        Use tech support chat.
        """
        useTechSupportChat(input: TechSupportChatInput!):
        TechSupportChatResponse @role(roles: [TECH_SUPPORT])

        """
        Use development chat.
        """
        useDevelopmentChat(input: DevelopmentChatInput!):
        DevelopmentChatResponse @role(roles: [DEVELOPER])
}

"""
Input type for tech support chat interactions.
"""
input TechSupportChatInput {
        userId: ID!
        message: String!
}

"""
Response type for tech support chat.
"""
type TechSupportChatResponse {
        success: Boolean!
        message: String
}

"""
Input type for development chat interactions.
"""
input DevelopmentChatInput {
        developerId: ID!
        message: String!
}

"""
Response type for development chat.
"""
type DevelopmentChatResponse {
        success: Boolean!
        message: String
}
```

4 Product backlog

Priority's (prio) value is decreasing, so the more important jobs have the lowest priority.

Estimation (est) is a numeric value attributed to the time estimation for the job completion. It's Higher value is 12, while the lowest is 0.

This section contains 3 subsections:

- The first subsection describes each backlog item as depicted in slide 39 of this document
- The second subsection describes each backlog item as depicted in slide 3 of this document
- The third subsection lists all those backlog items that cannot be described by an user story. Those items regard some documentation and setup actions that are still necessary and compose the first deliverable. Each item is described with both of the previous templates (adapted to the specific needs)

4.1 First definition of backlog item

Id	Name	User story	How to Demo	Prio	Est
1	Login	As a registered user I want to log into the platform so that I can access my roles privileges	Given that I'm at the home page, when I press on the login button and fill in my correct credentials I should get logged in	1	8
2	Logout	As a registered user I want to log out of the platform to prevent anyone to use my roles' privileges without my supervision	Given that I'm logged in, when I press on the logout button I should immediately be logged out and redirected to the home page	1	5
3	Credential recovery	As a registered user I want to be able to recover my credentials if I lose or forget them, in order to avoid losing my account over a single problem	Given that I'm trying to log in, I should be able to press the recovery button (near the credentials insertion boxes) and receive new credentials on the previously decided recovery channel	3	8
4	Register Account	As an unregistered user I want to register to the platform to keep track of my progresses and receive one or more roles	Given that I'm at the home page, when I press on the register button and fill in all the required credentials and info, my account should be created (with the required roles) and I should be automatically logged in	1	8
5	Delete account	As a registered user I want to be able to delete my account to delete my info from the platform	Given that I'm logged in, when I push the delete account button (in the options page) and confirm my choice (by filling in my password and pushing the according button), my account will be deleted along with all the related data	1	4
6	Modify core settings	As a registered user I want to be able to modify my core settings (username, password, recovery channel and personal info) in order to be able to change my mind after registration	Given that I'm in the core option page, when I push the modify button I'll able to refill each field in the core option list and, after pushing the save button, the changes will be saved	2	4

Id	Name	User story	How to Demo	Prio	Est
7	Modify secondary settings	As a registered user I want to be able to change my secondary settings (platform color theme, dashboard layout, text font and dimension, colorblind mode)	Given that I'm in the secondary option page, when I push the modify button I'll able to refill each field in the secondary option list and, after pushing the save button, the changes will be saved	4	4
8	Modify AI theming	As a student, AI supervisor or tech support I want to be able to modify the prompt for AI generated theming, in order to customize my experience (or test if the AI is working correctly)	Given that I'm on the dashboard page, when I push the modify theming button and fill in the new prompt, the theming for each exercise will change in accord with the new prompt	4	8
9	Access profile and statistics	As a registered user I want to be able to access my profile, in order to see if the saved info are correct and display all my progress on the platform	Given that I'm on the home page and I'm logged in, when I press on the button labeled with my username, I'll be redirected to the profile page	2	4
10	Change user role	As a registered user I want to be able to change the list of roles assigned to my account, in order to dynamically align with the list of actions I need to accomplish	Given that I'm at the profile page, when I push the manage roles button I will be redirected to the roles management page and I'll be able to add and remove roles	1	6
10.1	Add role	As a registered user I want to remove a role from my account, in order to remove access from useless actions	Given that I'm at the roles management page, when I press on a role and push the remove button, that role will be removed from the list and my account will not be associated with that role anymore	/	/
10.2	Remove role	As a registered user I want to add a role to my account, in order to have access to more actions	Given that I'm at the roles management page, when I push the add button and fill in all the required info, the required role will be add to the list and my account	/	/

Id	Name	User story	How to Demo	Prio	Est
11	Create course	As a publisher I want to be able to create new courses, in order to share my knowledge with the whole platform	Given that I'm on the dashboard page, when i press the add course button, I will be redirected to the course creation page. There I will be able to insert documents, videos, audio files, exercises and minigames. After I'll be finished and the save button is pushed, the course will be published on the platform	2	9
12	Modify course	As a professor, publisher, AI supervisor, course supervisor or system admins I want to modify an existing course (add, remove or modify content), in order to make it comply to what I need	Given that I'm on a course page (and have the right to modify it), when I press the modify button I will be redirected to the course creation page, there I will be able to add, modify or remove documents, videos, audio files, exercises and minigames. After I'll be finished and the save button is pushed, the course will be modified (each class based on the course will not be modified and keep the old version of the course)	2	6
13	Delete course	As a publisher, course supervisor or system admin I want to be able to delete an existing course, in order to remove from the platform unwanted info	Given that I'm on a course page (and have the right to delete it), when I push the delete button and confirm my choice, the course will be deleted from the platform (each class based on the course will be also deleted)	3	5
14	Archive course	As a publisher, course supervisor or system admin I want to be able to archive an existing course, in order to remove from the platform unwanted info, but keep existing classes alive	Given that I'm on a course page (and have the right to archive it), when I push the archive button and confirm my choice, the course will be archived and made invisible from the platform (each class based on the course will stay untouched)	5	4

Id	Name	User story	How to Demo	Prio	Est
15	View course	As a user I want to see any course and interact with its content	Given that I'm at the dashboard page, when I press on a course I will be redirected to its page. There the documents and reviews will be displayed, the video and audio files will be accessible, I'll be able to complete the exercises and play the minigames	3	3
16	Review course	As a registered user I want to be able to add a textual and numerical review to any course, in order to express my opinion on the quality of the selected course	Given that I'm on the course page, when I press the add review button, I'll be able to write a message and select a score among a list of options (integers from 0 to 10). After I'm finished and press the publish button, my review will be saved and displayed	5	6
17	Create class	As a professor I want to create an enclosed environment for my students, in order to keep track of their learning progress	Given that I'm on the dashboard page, when i press the add class button, I will be redirected to the class creation page. There I will be able to select the course I want to base my class on. Moreover I'll be able to insert a list of students' usernames, which will have access to the class. After I'll be finished and the save button is pushed, the class will be published on the platform	4	8
18	Modify class	As a professor or system admin I want to be able to change or update the course I'm basing my class on; in addition I want to modify the list of students, in order to have a dynamic experience with my students	Given that I'm at the selected class page, when I press the modify option I'll be redirected to the class creation page. There I will be able to change the course or update it to the newest version. Moreover I'll be able to modify the student list, by adding or removing usernames. After I'll be finished and the save button is pushed, the class will be updated	4	6

Id	Name	User story	How to Demo	Prio	Est
19	Terminate class	As a professor or system admin I want to fully erase a class and all its data from the platform, in order to remove old, unwanted or unused classes	Given that I'm at the class page, when I press the terminate button and confirm my choice, the class, all its data and statistics will be deleted	5	6
20	Archive class	As a professor or system admin I want to remove a class from the platform, but keep its statistics accessible, in order to still keep track of completed classes	Given that I'm at the class page, when I press the archive button and confirm my choice, the class will not be interactable anymore apart from the statistics	5	7
21	View class public info	As a registered user I want to be able to see the existing classes on the platform, in order to be able to ask for access	Given that I'm on the platform, every published class is listed and, by pressing on one of them, its owner, title and core course are displayed	4	4
22	Display class	As a student or professor I want to access the material of the course the class is based on, in order to interact with the class	Given that I'm on the dashboard page and am allowed to enter the class, when I press on it I'll be redirected to its page. There all the course components will be displayed and interactable. Moreover, I'll be able to see the username of every attendee. Furthermore I can see my statistics regarding the class' activities (as a professor I can see all students' statistics and the list of join requests)	4	5
23	Join class	As a student I want to ask a professor to join its class, in order to be part of that community	Given that I'm at the dashboard page, when I press on a class, the join button will be displayed. When I press that button my username will be add to the join request list of the class	4	4
24	Accept student	As a professor I want to be able to accept new students' request, in order to expand my class	Given that I'm on the class page, I can press the accept button next to the username of each username in the join list. After I press that, the student will be add to the students' list	4	5

Id	Name	User story	How to Demo	Prio	Est
25	Leave class	As a student I want to be able to leave a class, in order not to be stuck in an undesired class	Given that I'm in the class page, when I press on the leave button I'll be automatically deleted from the students' list. Moreover the professor will not be able to add my username to the list, unless I request to join	4	5
26	Publish article	As a publisher I want to be able to create new articles and decide whom to share them with, in order to share some smaller fragments of knowledge	Given that I'm on the dashboard page, when I press the add article button I'll be redirected to the article creation page. There I will be able to insert one document and/or one audio/video file. Moreover I can select a list of users who'll be able to see the article or just make it public. After I'm done and the save button is pushed the new article will be published on the platform	2	6
27	Modify article	As a publisher, course supervisor or system admin I want to be able to modify an existing article in order to make it better	Given that I'm on the article page, when I press the modify button I'll be redirected to the article creation page. There I'll be able to modify or substitute the content of the materials on the article. Moreover I'll be able to modify the visibility of the article and the users who can see it. After I'll be finished and the save button is pushed, the article will be modified	2	5
28	Delete article	As a publisher, course supervisor or system admin I want to be able to delete an article if it's incorrect or obsolete	When I'm on the article page and I press the delete button, after confirming my choice, the article will be removed entirely from the platform	3	5

Id	Name	User story	How to Demo	Prio	Est
29	Archive article	As a publisher, course supervisor or system admin I want to be able to archive the article previously published, in order to review it before remake it public	Given that I'm in the article page, when I press the archive button and confirm my choice, the article will become invisible to everyone apart from the owner	5	6
30	View article	As a registered user I want to be able to access the info published on the article, in order to acculturate myself	Given that I'm on the dashboard page, when I press on a visible article I'm redirected to its page. There I'll be able to read the document, see the video and/or watch the video	2	3
31	Review article	As a registered user I want to be able to publish a comment and/or a numeric value regarding an article, in order to express my opinion	Given that I'm in the article page, when I press the add review button I'll be able to write a message and select a score among a list of options (integers from 0 to 10). After I'm finished and press the publish button, my review will be saved and displayed	5	6
32	Create minigame	As a developer I want to be able to create a minigame, in order to satisfy the request of a publisher	Given that I'm on the dashboard page, when I press on the create minigame button I'll be redirected to the game creation engine. There the engine will allow me to model the aesthetic of the game, its mechanics, rules and all other components. After I'm done and the publish button is pressed, the minigame is published on the platform in the dashboard page	6	12
33	Modify minigame	As a developer, course supervisor, system admin or tech support I want to be able to modify an existing minigame to better it	Given that I'm on the game page, when I press the modify button I'll be redirected to the game creation engine. There I'll be able to modify every aspect of the minigame and, after I'll press the save button, all iterations of the minigame on the platform will be updated	6	11

Id	Name	User story	How to Demo	Prio	Est
34	Delete minigame	As a developer, course supervisor or system admin I want to be able to delete a minigame and all its iteration on the platform	Given that I'm on the minigame page, when I press the delete button and confirm my choice, the minigame is deleted from the platform both from the dashboard and all the courses containing it	7	6
35	Archive minigame	As a developer, course supervisor or system admin I want to be able to archive a minigame, in order to prevent further use of it in new courses, but not modify the existing ones	Given that I'm on the minigame page, when I press the archive button and confirm my choice, the minigame is deleted from the dashboard and cannot be assigned to any new course	7	7
36	Add minigame from dev	As a developer I want to be able to add the minigame I created to the required course, in order to complete my work	Given that I'm in the game page, when I press the add to course button the list of courses I have access to will be displayed. When I press on one of them and confirm the choice, the game will be add to the selected course	6	5
37	Add minigame from observer	As a publisher, course supervisor or system admin I want to be able to add a minigame to a specific course, in order to make such course more interactive	Given that I'm at the course page, when I press the add minigame button a list of available minigames is opened. When I press on one of them and confirm my choice, such minigame is add to the course	6	5
38	View and play minigame	As a user I want to be able to play the available minigames, both on the dashboard and the courses, in order to learn by doing	Given that I'm on the dashboard or on a course page, when I press on the name of the minigame I'll be redirected to its page. There I'll be able to interact and play with the game as intended by the creator	6	10

Id	Name	User story	How to Demo	Prio	Est
39	Pay developer	As a publisher I want to be able to pay the developer I hired when they complete the minigame I commissioned, in order to delegate that part of the course creation	Given that I'm at the dashboard page, when I press on the pay button I'll be redirected to the payment page. There I'll be able to see a list of developers and select the person I want to pay. After that I'll select the payment method I want to use and follow the iter associated with the external method selected	6	7
40	Use tech support chat	As a registered user I want to internally discuss a problem of the platform, in order to solve them and better the platform experience	Given that I'm on any page of the platform, when I press the tech chat button, the chat window will be displayed on top of the page I'm in. As a normal user I'll be able to send messages to the tech supports. As a tech support I'll be able to respond to each message (coming from different channels depending on the sender) to help solve the problems	5	9
41	Use dev chat	As a publisher or developer I want to internally talk about the creation of a minigame, in order to have a specialized person develop them	Given that I'm on the dashboard page, when I press on the dev chat button I'll be redirected to the chat page. There I'll have a channel for each user I have talked to and a list of existing publishers and developers to select and start a chat with	6	9
42	Search material	As a user I want to be able to search for the info I want, in order to find only the stuff I'm interested in	Given that I'm on the dashboard page, when I press on the search button I'll be able to insert the words and select the tags I want to search with. After that the dashboard's content will be reordered based on accordance with the required parameters	2	7
43	Remove review	As a system admin I want to be able to remove reviews of contents, in order to moderate the environment and avoid toxic behaviours	Given that I'm on any reviewable content, when I press on the delete button next to a review and confirm my choice, such review will be eliminated from the platform	5	5

4.2 Second definition of backlog item

4.2.1 Login

- ID: 1
- Title: Login
- User story: As a registered user I want to log into the platform so that I can access my roles privileges
- Acceptance criteria: The user is able to access their account with their credentials
- Priority: 1
- Estimate: 8
- Status: to-do
- Sprint assignment: S2
- Theme: Account management
- Notes:

4.2.2 Logout

- ID: 2
- Title: Logout
- User story: As a registered user I want to log out of the platform to prevent anyone to use my roles' privileges without my supervision
- Acceptance criteria: The user is able to return to anonymous mode after being logged in
- Priority: 1
- Estimate: 5
- Status: to-do
- Sprint assignment: S2
- Theme: Account management
- Notes:

4.2.3 Credential recovery

- ID: 3
- Title: Credential recovery
- User story: As a registered user I want to be able to recover my credentials if I lose or forget them, in order to avoid losing my account over a single problem
- Acceptance criteria: The new credentials can be shipped through every recovery channel with a sufficiently small time delay
- Priority: 3
- Estimate: 8
- Status: to-do

- Sprint assignment: S4 or later
- Theme: Account management
- Notes:

4.2.4 Register Account

- ID: 4
- Title: Register Account
- User story: As an unregistered user I want to register to the platform to keep track of my progresses and receive one or more roles
- Acceptance criteria: Any new account is correctly registered in the DB with its associated roles
- Priority: 1
- Estimate: 8
- Status: to-do
- Sprint assignment: S2
- Theme: Account management
- Notes:

4.2.5 Delete account

- ID: 5
- Title: Delete account
- User story: As a registered user I want to be able to delete my account to delete my info from the platform
- Acceptance criteria: All the info regarding any account can be completely removed from the DB
- Priority: 1
- Estimate: 4
- Status: to-do
- Sprint assignment: S2
- Theme: Account management
- Notes:

4.2.6 Modify core settings

- ID: 6
- Title: Modify core settings
- User story: As a registered user I want to be able to modify my core settings (username, password, recovery channel and personal info) in order to be able to change my mind after registration
- Acceptance criteria: The info in the DB can be easily be modified only be the owner of a certain account

- Priority: 2
- Estimate: 4
- Status: to-do
- Sprint assignment: S3
- Theme: Account management
- Notes:

4.2.7 Modify secondary settings

- ID: 7
- Title: Modify secondary settings
- User story: As a registered user I want to be able to change my secondary settings (platform color theme, dashboard layout, text font and dimension, colorblind mode)
- Acceptance criteria: The platform gets modified in accordance with the new info in a relatively short amount of time
- Priority: 4
- Estimate: 4
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Account management
- Notes:

4.2.8 Modify AI theming

- ID: 8
- Title: Modify AI theming
- User story: As a student, AI supervisor or tech support I want to be able to modify the prompt for AI generated theming, in order to customize my experience (or test if the AI is working correctly)
- Acceptance criteria: The exercises theming is modified in accordance with the new prompt in an acceptable amount of time
- Priority: 4
- Estimate: 8
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Account management
- Notes:

4.2.9 Access profile and statistics

- ID: 9
- Title: Access profile and statistics
- User story: As a registered user I want to be able to access my profile, in order to see if the saved info are correct and display all my progress on the platform
- Acceptance criteria: The user info can be displayed whenever the user wants easily
- Priority: 2
- Estimate: 4
- Status: to-do
- Sprint assignment: S2/S3
- Theme: Account management
- Notes:

4.2.10 Change user role

- ID: 10
- Title: Change user role
- User story: As a registered user I want to be able to change the list of roles assigned to my account, in order to dynamically align with the list of actions I need to accomplish
- Acceptance criteria: The roles can be changed (add or remove) with the correct security and accessibility
- Priority: 1
- Estimate: 6
- Status: to-do
- Sprint assignment: S2
- Theme: Account management
- Notes:

4.2.11 Create course

- ID: 11
- Title: Create course
- User story: As a publisher I want to be able to create new courses, in order to share my knowledge with the whole platform
- Acceptance criteria: A new course can be modeled and published how and when a publisher wants
- Priority: 2
- Estimate: 9
- Status: to-do
- Sprint assignment: S3
- Theme: Course management
- Notes:

4.2.12 Modify course

- ID: 12
- Title: Modify course
- User story: As a professor, publisher, AI supervisor, course supervisor or system admins I want to modify an existing course (add, remove or modify content), in order to make it comply to what I need
- Acceptance criteria: The user is able to modify the content of a course they have the right to whenever they want
- Priority: 2
- Estimate: 6
- Status: to-do
- Sprint assignment: S3
- Theme: Course management
- Notes:

4.2.13 Delete course

- ID: 13
- Title: Delete course
- User story: As a publisher, course supervisor or system admin I want to be able to delete an existing course, in order to remove from the platform unwanted info
- Acceptance criteria: Every trace of the selected course is removed from the DB
- Priority: 3
- Estimate: 5
- Status: to-do
- Sprint assignment: S3/S4
- Theme: Course management
- Notes:

4.2.14 Archive course

- ID: 14
- Title: Archive course
- User story: As a publisher, course supervisor or system admin I want to be able to archive an existing course, in order to remove from the platform unwanted info, but keep existing classes alive
- Acceptance criteria: The course is not accessible anymore from the platform apart for the classes containing it
- Priority: 5
- Estimate: 4
- Status: to-do

- Sprint assignment: S5 or later
- Theme: Course management
- Notes:

4.2.15 View course

- ID: 15
- Title: View course
- User story: As a user I want to see any course and interact with its content
- Acceptance criteria: Any public course is accessible from the platform and its content can be interacted without bugs
- Priority: 3
- Estimate: 3
- Status: to-do
- Sprint assignment: S3
- Theme: Course management
- Notes:

4.2.16 Review course

- ID: 16
- Title: Review course
- User story: As a registered user I want to be able to add a textual and numerical review to any course, in order to express my opinion on the quality of the selected course
- Acceptance criteria: The reviews on courses can be published and visualized easily
- Priority: 5
- Estimate: 6
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Course management
- Notes:

4.2.17 Create class

- ID: 17
- Title: Create class
- User story: As a professor I want to create an enclosed environment for my students, in order to keep track of their learning progress
- Acceptance criteria: A new class environment can easily be based on a course and published for the given student set to see

- Priority: 4
- Estimate: 8
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.18 Modify class

- ID: 18
- Title: Modify class
- User story: As a professor or system admin I want to be able to change or update the course I'm basing my class on; in addition I want to modify the list of students, in order to have a dynamic experience with my students
- Acceptance criteria: The class' course and list of students can be modified whenever the people with the right to want and the modification is saved in the DB in a short time
- Priority: 4
- Estimate: 6
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.19 Terminate class

- ID: 19
- Title: Terminate class
- User story: As a professor or system admin I want to fully erase a class and all its data from the platform, in order to remove old, unwanted or unused classes
- Acceptance criteria: The class and all its data are entirely removed from the platform and DB
- Priority: 5
- Estimate: 6
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Class management
- Notes:

4.2.20 Archive class

- ID: 20
- Title: Archive class
- User story: As a professor or system admin I want to remove a class from the platform, but keep its statistics accessible, in order to still keep track of completed classes
- Acceptance criteria: Classes and their data cannot be accessed anymore from the platform apart from the classes' statistics
- Priority: 5
- Estimate: 7
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Class management
- Notes:

4.2.21 View class public info

- ID: 21
- Title: View class public info
- User story: As a registered user I want to be able to see the existing classes on the platform, in order to be able to ask for access
- Acceptance criteria: The classes' public info are displayed every time a registered user selects a public class entry
- Priority: 4
- Estimate: 4
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.22 Display class

- ID: 22
- Title: Display class
- User story: As a student or professor I want to access the material of the course the class is based on, in order to interact with the class
- Acceptance criteria: The class' attendees can access all the material contained in it and interact with the games/exercises
- Priority: 4
- Estimate: 5
- Status: to-do

- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.23 Join class

- ID: 23
- Title: Join class
- User story: As a student I want to ask a professor to join its class, in order to be part of that community
- Acceptance criteria: The name of the student is add to the class' list of join request on the class and such list is updated in a short time
- Priority: 4
- Estimate: 4
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.24 Accept student

- ID: 24
- Title: Accept student
- User story: As a professor I want to be able to accept new students' request, in order to expand my class
- Acceptance criteria: The selected student is add to the class' list of students, such list is updated and the student is granted access to the class' resources
- Priority: 4
- Estimate: 5
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.25 Leave class

- ID: 25
- Title: Leave class
- User story: As a student I want to be able to leave a class, in order not to be stuck in an undesired class
- Acceptance criteria: The student leaves a class and, if the professor tries to re-invite the student, they are not add to the join list for the left class
- Priority: 4
- Estimate: 5
- Status: to-do
- Sprint assignment: S4 or later
- Theme: Class management
- Notes:

4.2.26 Publish article

- ID: 26
- Title: Publish article
- User story: As a publisher I want to be able to create new articles and decide whom to share them with, in order to share some smaller fragments of knowledge
- Acceptance criteria: the article is modeled and published however the publisher wants and is accessible to whomever they want
- Priority: 2
- Estimate: 6
- Status: to-do
- Sprint assignment: S2/S3
- Theme: Article management
- Notes:

4.2.27 Modify article

- ID: 27
- Title: Modify article
- User story: As a publisher, course supervisor or system admin I want to be able to modify an existing article in order to make it better
- Acceptance criteria: Every modification to the article's content or visibility is saved and applied in a short range of time
- Priority: 2
- Estimate: 5
- Status: to-do

- Sprint assignment: S2/S3
- Theme: Article management
- Notes:

4.2.28 Delete article

- ID: 28
- Title: Delete article
- User story: As a publisher, course supervisor or system admin I want to be able to delete an article if it's incorrect or obsolete
- Acceptance criteria: The article is entirely removed from the platform and DB
- Priority: 3
- Estimate: 5
- Status: to-do
- Sprint assignment: S3 or later
- Theme: Article management
- Notes:

4.2.29 Archive article

- ID: 29
- Title: Archive article
- User story: As a publisher, course supervisor or system admin I want to be able to archive the article previously published, in order to review it before remake it public
- Acceptance criteria: The article becomes inaccessible for anyone other than its owner
- Priority: 5
- Estimate: 6
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Article management
- Notes:

4.2.30 View article

- ID: 30
- Title: View article
- User story: As a registered user I want to be able to access the info published on the article, in order to acculturate myself
- Acceptance criteria: The article page is accessible by everyone on the platform
- Priority: 2

- Estimate: 3
- Status: to-do
- Sprint assignment: S2/S3
- Theme: Article management
- Notes:

4.2.31 Review article

- ID: 31
- Title: Review article
- User story: As a registered user I want to be able to publish a comment and/or a numeric value regarding an article, in order to express my opinion
- Acceptance criteria: The user's review is saved and published on the platform for anyone to see
- Priority: 5
- Estimate: 6
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Article management
- Notes:

4.2.32 Create minigame

- ID: 32
- Title: Create minigame
- User story: As a developer I want to be able to create a minigame, in order to satisfy the request of a publisher
- Acceptance criteria: The minigame development environment is accessible and functional to the developer. Moreover the games created are functional and bug-less
- Priority: 6
- Estimate: 12
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.33 Modify minigame

- ID: 33
- Title: Modify minigame
- User story: As a developer, course supervisor, system admin or tech support I want to be able to modify an existing minigame to better it
- Acceptance criteria: The modification through the development environment is saved on the entire platform
- Priority: 6
- Estimate: 11
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.34 Delete minigame

- ID: 34
- Title: Delete minigame
- User story: As a developer, course supervisor or system admin I want to be able to delete a minigame and all its iteration on the platform
- Acceptance criteria: All the minigame's presence is erased from the entire platform and DB
- Priority: 7
- Estimate: 6
- Status: to-do
- Sprint assignment: S7 or later
- Theme: Minigame management
- Notes:

4.2.35 Archive minigame

- ID: 35
- Title: Archive minigame
- User story: As a developer, course supervisor or system admin I want to be able to archive a minigame, in order to prevent further use of it in new courses, but not modify the existing ones
- Acceptance criteria: The minigame is inaccessible apart through the courses already containing it
- Priority: 7
- Estimate: 7
- Status: to-do
- Sprint assignment: S7 or later
- Theme: Minigame management
- Notes:

4.2.36 Add minigame from dev

- ID: 36
- Title: Add minigame from dev
- User story: As a developer I want to be able to add the minigame I created to the required course, in order to complete my work
- Acceptance criteria: The minigame is properly linked to the course page and when a user presses on it the page opens correctly
- Priority: 6
- Estimate: 5
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.37 Add minigame from observer

- ID: 37
- Title: Add minigame from observer
- User story: As a publisher, course supervisor or system admin I want to be able to add a minigame to a specific course, in order to make such course more interactive
- Acceptance criteria: The minigame is properly linked to the course page and when a user presses on it the page opens correctly
- Priority: 6
- Estimate: 5
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.38 View and play minigame

- ID: 38
- Title: View and play minigame
- User story: As a user I want to be able to play the available minigames, both on the dashboard and the courses, in order to learn by doing
- Acceptance criteria: The minigame is correctly visualized and works properly
- Priority: 6
- Estimate: 10
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.39 Pay developer

- ID: 39
- Title: Pay developer
- User story: As a publisher I want to be able to pay the developer I hired when they complete the minigame I commissioned, in order to delegate that part of the course creation
- Acceptance criteria: The external payment system works properly to pay the developers
- Priority: 6
- Estimate: 7
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Minigame management
- Notes:

4.2.40 Use tech support chat

- ID: 40
- Title: Use tech support chat
- User story: As a registered user I want to internally discuss a problem of the platform, in order to solve them and better the platform experience
- Acceptance criteria: The chat system works properly and can be relied upon to report bugs
- Priority: 5
- Estimate: 9
- Status: to-do
- Sprint assignment: S5 or later
- Theme: Chat systems
- Notes:

4.2.41 Use dev chat

- ID: 41
- Title: Use dev chat
- User story: As a publisher or developer I want to internally talk about the creation of a minigame, in order to have a specialized person develop them
- Acceptance criteria: The chat system works properly and can be relied upon to discuss the development of a minigame
- Priority: 6
- Estimate: 9
- Status: to-do
- Sprint assignment: S6 or later
- Theme: Chat systems
- Notes:

4.2.42 Search material

- ID: 42
- Title: Search material
- User story: As a user I want to be able to search for the info I want, in order to find only the stuff I'm interested in
- Acceptance criteria: The search engine filters and orders elements correctly on the dashboard
- Priority: 2
- Estimate: 7
- Status: to-do
- Sprint assignment: S2/S3
- Theme: General purpose
- Notes:

4.2.43 Remove review

- ID: 43
- Title: Remove review
- User story: As a system admin I want to be able to remove reviews of contents, in order to moderate the environment and avoid toxic behaviours
- Acceptance criteria: The reviews are entirely removed from the platform and DB
- Priority: 5
- Estimate: 5
- Status: to-do
- Sprint assignment: S5 or later
- Theme: General purpose
- Notes:

4.3 Documents and not implement entries

4.3.1 Definition of Done

Id	Name	Description	Prio	Est
44	Definition of Done	Creation of Definition of Done document, containing every criteria regarding the code of conduct and when a task should be considered as completed	0	2

- ID: 44
- Title: Definition of Done
- Description: Creation of Definition of Done document, containing every criteria regarding the code of conduct and when a task should be considered as completed
- Acceptance criteria: The document is considered complete and the description is accepted by the whole team
- Priority: 0
- Estimate: 2
- Status: done
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.2 Environment setup

Id	Name	Description	Prio	Est
45	Environment setup	Need to setup the environment for the platform	0	3

- ID: 45
- Title: Environment setup
- Description: Need to setup the environment for the platform
- Acceptance criteria: The environment is setup and ready for the implementation of the platform
- Priority: 0
- Estimate: 3
- Status: done
- Sprint assignment: S1
- Theme: Setup
- Notes:

4.3.3 Architecture description

Id	Name	Description	Prio	Est
46	Architecture description	Graphical description of the entire project and its inner workings	0	2

- ID: 46
- Title: Architecture description
- Description: Graphical description of the entire project and its inner workings
- Acceptance criteria: The schema is graphically appealing and completely describes the project structure
- Priority: 0
- Estimate: 2
- Status: done
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.4 Definition of Tests

Id	Name	Description	Prio	Est
47	Definition of Tests	Textual document containing the logic of the tests	0	2

- ID: 47
- Title: Definition of Tests
- Description: Textual document containing the logic of the tests
- Acceptance criteria: The test are depicted completely and are considered satisfactory for the customer
- Priority: 0
- Estimate: 2
- Status: done
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.5 Manual Product Backlog definition

Id	Name	Description	Prio	Est
48	Manual Product Backlog definition	Creation of this document, which contains all the info regarding the project's components and their info	0	4

- ID: 48
- Title: Manual Product Backlog definition
- Description: Creation of this document, which contains all the info regarding the project's components and their info
- Acceptance criteria: The document is complete with every item that is considered as part of the project and their info is exhaustive
- Priority: 0
- Estimate: 4
- Status: in progress
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.6 Git strategy

Id	Name	Description	Prio	Est
49	Git strategy	Creation of a set of rules, procedures and naming conventions with a visual example scenario to efficently and coherently work with git	0	3

- ID: 49
- Title: Git strategy
- Description: Decision of set of rules for clear and coherent usage of Git, with also the graphical representation
- Acceptance criteria: The image is clear and graphically appealing
- Priority: 0
- Estimate: 3
- Status: in progress
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.7 API definition

Id	Name	Description	Prio	Est
50	API definition	Define APIs in natural language	0	5

- ID: 50
- Title: API definition
- Description: Define APIs in natural language
- Acceptance criteria: The APIs are clear as how they work
- Priority: 0
- Estimate: 5
- Status: in progress
- Sprint assignment: S2
- Theme: Documents
- Notes:

4.3.8 Burndown Chart Automation

Id	Name	Description	Prio	Est
51	Burndown Chart Automation	Being able to automate the generation of a burndown chart from the GitHub environment	0	2

- ID: 51
- Title: Burndown Chart Automation
- Description: Being able to automate the generation of a burndown chart from the GitHub environment
- Acceptance criteria: Being able to automate the generation of a burndown chart from the GitHub environment
- Priority: 0
- Estimate: 2
- Status: in progress
- Sprint assignment: S1
- Theme: Automation
- Notes:

4.3.9 Sprint 1 setup

Id	Name	Description	Prio	Est
52	Sprint 1 setup	Decision and definition of all the tasks that must be accomplished in sprint 1	0	3

- ID: 52
- Title: Sprint 1 setup
- Description: Decision and definition of all the tasks that must be accomplished in sprint 1
- Acceptance criteria: The whole team has a clear idea of what must be completed
- Priority: 0
- Estimate: 3
- Status: done
- Sprint assignment: S1
- Theme: Documents
- Notes:

4.3.10 Sprint 1 Review and Retrospective

Id	Name	Description	Prio	Est
53	Sprint 1 Review and Retrospective	Complete review and retrospective about sprint 1	0	2

- ID: 53
- Title: Sprint 1 Review and Retrospective
- Description: Complete review and retrospective about sprint 1
- Acceptance criteria: These reports are clearly defined and complete
- Priority: 0
- Estimate: 2
- Status: done
- Sprint assignment: S1
- Theme: Documents
- Notes:

5 Sprint 1 backlog

Priorities may vary from the product backlog as they are considered inside the single sprint

Id	Name	Description	Acceptance criteria	Prio	Est	Volunteers
44	Definition of Done	Creation of Definition of Done document, containing every criteria regarding the code of conduct and when a task should be considered as completed	The document is considered complete and the description is accepted by the whole team	1	2	Lorenzo N Gabriele
45	Environment setup	Need to setup the environment for the platform	The environment is setup and ready for the implementation of the platform	1	3	Ahn Tu
46	Architecture description	Graphical description of the entire project and its inner workings	The schema is graphically appealing and completely describes the project structure	1	2	Lorenzo N Lorenzo C Ahn Tu
47	Definition of Tests	Textual document containing the logic of the tests	The test are depicted completely and are considered satisfactory for the customer	2	2	Ahn Tu
48	Product Backlog definition	Creation of this document, which contains all the info regarding the project's components and their info	The document is complete with every item that is considered as part of the project and their info is exhaustive	0	4	Gabriele

Id	Name	Description	Acceptance criteria	Prio	Est	Volunteers
49	Git strategy	Creation of a set of rules, procedures and naming conventions with a visual example scenario to efficently and coherently work with git	The image is clear and graphically appealing	3	3	Lorenzo N Lorenzo C
51	Burndown Chart Automation	Being able to automate the generation of a burndown chart from the GitHub environment	Being able to automate the generation of a burndown chart from the GitHub environment	4	2	Lorenzo C Gabriele Ahn Tu
52	Sprint 1 setup	Decision and definition of all the tasks that must be accomplished in sprint 1	The whole team has a clear idea of what must be completed	0	3	Lorenzo N Gabriele Lorenzo C Ahn Tu
53	Sprint 1 Review and Retrospective	Complete review and retrospective about sprint 1	These reports are clearly defined and complete	5	2	Lorenzo N Gabriele Lorenzo C Ahn Tu

6 Definition of tests

6.1 Overview

6.1.1 Importance of Testing

Testing plays a critical role in ensuring the functionality, usability, and security of RadiantIQ. As an platform, it must not only meet industry standards for software quality but also provide a reliable and efficient learning experience for our users. Comprehensive testing helps identify and address issues early in the development process, reducing critical issues impacting the user experience post-deployment.

6.1.2 Scope

This document outlines the testing strategy for RadianIQ, detailing the objectives, testing types, approach, test cases, test environment, test execution process, risks, and mitigation strategies. It serves as a guideline for the testing team, developers, and stakeholders involved in the development and quality assurance of the application. RadiantIQ test development will be based on the outlined below and encourage to use mixed testing methods to ensure the quality of the platform.

6.2 Testing Objectives

6.2.1 Goals of the Testing Strategy

The primary objective is to ensure that the platform meets the highest standards of quality, reliability, and security. Specifically, it aims to:

- Verify the functionality of all features: Ensure that each function of the web application, including course and class management, user registration, content delivery, and communication features, works as intended without any errors or unexpected behavior.
- Usability and accessibility: Evaluate the user-friendliness and accessibility of the platform to ensure that users can navigate the application easily and perform their tasks efficiently.
- Ensure compatibility across different environments: Confirm that RadianIQ is compatible with various browsers (e.g., Chrome, Firefox, Safari), devices (e.g., desktops, laptops, tablets, smartphones), and operating systems (e.g., Windows, macOS, iOS, Android).
- Evaluate performance: Test the responsiveness and scalability of the application to ensure that it can handle multiple users accessing the platform simultaneously and perform well under high peak.
- Identify and mitigate security vulnerabilities: Conduct thorough security testing to identify potential vulnerabilities, such as data breaches, unauthorized access, and injection attacks, and implement measures to mitigate these risks and protect user data.
- Ensure integration between components: Validate the interaction between different components and modules of the platform to ensure seamless functionality and data exchange.

6.3 Testing Types

6.3.1 Functional Testing (Black-box Testing)

Functional testing ensures that each function of RadianIQ works as expected. This involves testing individual features and functionalities to verify that they meet the specified requirements, including:

- Course Management: Testing the creation, editing, and deletion of courses, including features such as adding course materials, setting up assessments, and managing enrollments.

- Class Management: Testing the creation, editing, and deletion of classes, including features such as creating class sessions, managing student registrations, tracking student progress, and grading assignments.
- User Registration and Authentication: Testing the registration process for all the user roles, as well as the login and authentication mechanisms to ensure secure access to the platform.
- Content Delivery: Testing the delivery of course content, including multimedia materials, documents, mini games, and lectures, to ensure they are accessible and functional.
- Communication Features: Testing messaging systems, discussion forums, and collaboration tools to ensure smooth communication and interaction between users.

6.3.2 Structural Testing (White-box Testing)

Structural testing, or white-box testing, involves examining the internal structure and code of the e-learning web application to validate its logic, algorithms, and internal pathways. This testing type ensures the reliability, security, and maintainability of the application's underlying codebase. Key aspects to test include:

- Code Coverage: Verify that all code paths, including branches, loops, and conditionals, are executed and tested to achieve high code coverage.
- Data Validation: Validate input data to prevent security vulnerabilities such as SQL injection, cross-site scripting (XSS), and data manipulation attacks.
- Error Handling: Ensure robust error handling mechanisms are in place to gracefully handle exceptions, invalid inputs, and unexpected scenarios without crashing the application.
- Performance Optimization: Analyze code performance to identify and optimize inefficient algorithms, database queries, and resource-intensive operations for improved scalability and responsiveness.

6.3.3 Usability Testing

Usability testing focuses on assessing the user-friendliness and accessibility of the platform. This involves evaluating the interface design, navigation flow, and overall user experience to ensure that users can easily accomplish their tasks and achieve their learning objectives. Usability testing includes:

- Navigation Testing: Evaluating the navigation flow within the application to ensure that users can easily find and access the desired features and content.
- Accessibility Testing: Ensuring that the platform is accessible to users with disabilities, including compliance with accessibility standards such as WCAG (Web Content Accessibility Guidelines).
- User Feedback Analysis: Gathering feedback from users through surveys, interviews, or usability testing sessions to identify areas for improvement in the user interface and experience.

6.3.4 Compatibility Testing

Compatibility testing ensures that RadianIQ is compatible across different browsers, devices, and operating systems. This involves testing the application on various combinations of browsers (e.g., Chrome, Firefox, Safari), devices (e.g., desktops, laptops, tablets, smartphones), and operating systems (e.g., Windows, macOS, iOS, Android) to ensure consistent performance and functionality.

6.3.5 Performance Testing

Performance testing evaluates the responsiveness and scalability of the platform under various loads. This involves testing the application's performance metrics such as response time, throughput, and resource utilization to ensure optimal performance under normal and peak usage conditions. Performance testing includes:

- Load Testing: Simulating multiple users accessing the platform simultaneously to assess its response time and behavior under heavy loads.
- Stress Testing: Pushing the system beyond its normal capacity to identify its breaking point and evaluate its ability to recover under stress.
- Scalability Testing: Testing the application's ability to scale resources dynamically to accommodate increasing user loads without degradation in performance.

6.3.6 Security Testing

Security testing aims to identify and mitigate potential security vulnerabilities in the platform. This involves testing the application for vulnerabilities such as data breaches, unauthorized access, injection attacks, and XSS (cross-site scripting) attacks. Security testing includes:

- Vulnerability Assessment: Identifying potential security vulnerabilities through automated scanning tools and manual code reviews.
- Penetration Testing: Simulating real-world attacks to exploit vulnerabilities and assess the effectiveness of security controls and countermeasures.
- Data Protection Testing: Ensuring that sensitive user data such as personal information, grades, and assessment results are securely stored, transmitted, and accessed according to privacy regulations and best practices.

6.4 Testing Method Approach

6.4.1 V-Model Testing

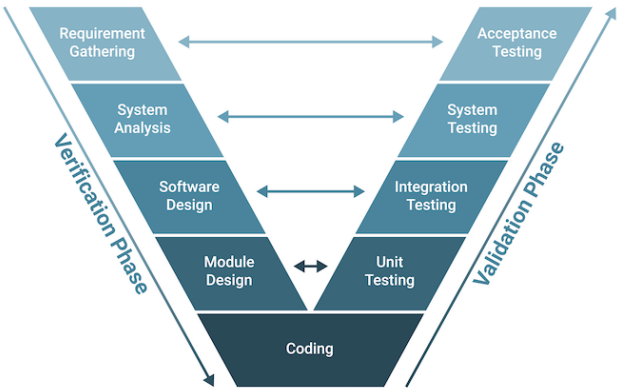

Figure 4: V-model testing

The V-Model testing approach aligns testing activities with corresponding development phases, emphasizing the relationship between requirements, design, implementation, and testing. In the V-Model, each phase of the development lifecycle has a corresponding testing phase, ensuring comprehensive validation at each stage. The V-Model testing process includes:

- Acceptance Testing (Requirements Gathering): During the requirements gathering phase, acceptance testing focuses on validating the gathered requirements to ensure they are clear, complete, and consistent. This involves collaborating with stakeholders to define acceptance criteria and validate that the requirements accurately capture the needs of end-users and align with business objectives.
- System Testing (System Analysis): In the system analysis phase, system testing verifies the complete integrated system against the specified requirements and design specifications. This includes testing the system as a whole to ensure that all components function correctly together and meet the intended functionality, performance, and usability criteria.
- Integration Testing (Software Design): Integration testing validates the interaction and integration between individual software modules or components. It ensures that modules interact correctly, exchange data seamlessly, and function as expected within the larger system architecture. Integration testing verifies that the software design, including interfaces and dependencies, is implemented correctly and that modules work together harmoniously. modules.
- Unit Testing (Module Design): Unit testing focuses on validating the functionality of individual software modules or components in isolation. It verifies that each module performs its intended function according to the specified design and requirements. Unit tests are designed to test specific inputs, outputs, and internal logic of modules, ensuring that each unit operates correctly and meets its design specifications.

6.4.2 Manual Testing

Manual testing involves human testers executing test cases and evaluating the application's behavior based on predefined criteria. This approach allows testers to identify visual inconsistencies, usability issues, and other aspects that may be challenging to automate. The manual testing process includes:

- Test Case Creation: Developing detailed test cases based on functional requirements and user stories.
- Test Execution: Performing manual tests according to the test cases, documenting observations, and verifying expected outcomes.
- Exploratory Testing: Conducting ad-hoc testing to explore the application and uncover potential issues that may not be covered by predefined test cases.
- Usability Testing: Engaging real users to evaluate the application's usability, accessibility, and overall user experience.
- Regression Testing: Repeating tests to ensure that new features or changes do not adversely affect existing functionalities.

6.4.3 Automated Testing

Automated testing involves using software tools and frameworks to automate the execution of test cases, thereby increasing efficiency, repeatability, and coverage. This approach is particularly useful for repetitive tasks and regression testing. The automated testing process includes:

- Test Script Development: Writing test scripts using automated testing tools such as Selenium, Cypress, or TestComplete to simulate user interactions and verify expected behaviors.
- Test Suite Creation: Organizing test scripts into test suites based on functional areas or features for efficient execution and maintenance.
- Continuous Integration (CI) Integration: Integrating automated tests into the CI/CD pipeline to automatically trigger tests upon code changes and provide rapid feedback to developers.

- Cross-Browser and Cross-Platform Testing: Executing tests across different browsers, devices, and operating systems to ensure compatibility and consistent behavior.
- Performance Testing Automation: Utilizing tools like JMeter or Gatling for automating performance tests to simulate various load scenarios and analyze system performance metrics.

6.4.4 Regression Testing

Regression testing ensures that new changes or enhancements do not introduce unintended side effects or break existing functionalities. The regression testing process includes:

- Test Case Maintenance: Updating existing test cases and creating new ones to accommodate changes in the application.
- Test Suite Prioritization: Prioritizing test cases based on their criticality and impact on the application to optimize testing efforts.
- Automated Regression Testing: Automating repetitive regression tests to ensure consistent and efficient validation of core functionalities.
- Manual Regression Testing: Performing manual regression tests for areas that are difficult to automate or require human judgment.
- Version Control Integration: Integrating regression tests with version control systems to track changes and ensure test coverage across different software versions.

6.5 Test Environment

6.5.1 Hardware Specifications

The test environment setup for RadianIQ should include hardware specifications that closely resemble the production environment to ensure accurate testing results. The hardware specifications may include:

- Server Configuration: Specifications for Render.com, server hosting the web application, including CPU, RAM, and storage capacity.
- Client Devices: Specifications for client devices used to access the application, such as desktop computers, laptops, tablets, and smartphones.
- Networking Equipment: Details of networking hardware such as routers, switches, and firewalls that may impact network performance and connectivity.

6.5.2 Software Dependencies

The test environment should replicate the software dependencies of the production environment to ensure compatibility and accurate testing. This includes:

- Operating Systems: Versions of operating systems supported by the platform.
- Web Browsers: Versions of web browsers supported by the application, including Chrome, Firefox, Safari, and Edge.
- Database Systems: Versions of database management systems (DBMS) used by the application. In this case, MongoDB.
- Server Software: Versions of web servers, application servers, and other server software required to run the application.

6.5.3 Network Configurations

Network configurations play an important role in testing the performance and reliability. The test environment should simulate real-world network conditions to evaluate the application's behavior under different scenarios. This includes:

- Internet Connectivity: Ensure stable internet connectivity with sufficient bandwidth to support concurrent user interactions.
- Firewall and Security Settings: Configure network security settings to simulate firewall rules and security protocols that may impact application access and communication.
- Load Balancing: Implement load balancing configurations to distribute traffic across multiple servers and assess scalability and performance under heavy loads.
- Latency and Packet Loss: Introduce latency and packet loss to simulate network congestion and assess application responsiveness and resilience.

6.5.4 Data Sets for Testing

Data sets used for testing should represent realistic scenarios and volumes to validate the performance, scalability, and reliability of the platform. This includes:

- Sample Courses and Content: Populate the test environment with a variety of sample courses, lectures, mini games, and multimedia content to simulate real usage scenarios.
- User Data: Create test user accounts with different roles and varying levels of access permissions to test user registration, authentication, and authorization functionalities.
- Test Data Generation: Use data generation tools or scripts to create large volumes of test data to simulate realistic user interactions, such as enrollment in courses, participation in discussions, and submission of assignments.

6.6 Bug Reporting and Tracking

Bug reporting and tracking are essential aspects of the test execution process, enabling testers to document and manage issues effectively. The bug reporting and tracking process includes:

- Defect Identification: Identify and document defects discovered during test execution, including detailed descriptions, steps to reproduce, and screenshots or logs.
- Defect Prioritization: Prioritize defects based on severity, impact on functionality, and business priority to focus on resolving critical issues first.
- Defect Management: Assign, track, and manage defects using a centralized defect tracking system or bug management tool to ensure transparency, accountability, and traceability throughout the defect lifecycle.
- Defect Resolution: Collaborate with development teams to investigate, triage, and resolve reported defects promptly, following established workflows and communication channels.

6.7 Test cases template

Providing here the base test cases table, where each iteration will define more test cases.

N.	Description	Test case data	Precondition	Dep.	Expected result	Actual result	Notes
0	Login with unregisted user	Go to <URL_LOGIN> insert username and password			A message is displayed "Invalid username or password". The login page is reload.		
1.1	Register new user successfully	Go to <URL_REG> and insert firstname, lastname, email, username and password			A message is displayed "User created successfully". The dashboard is loaded.		
1.2	Register new user failed	Go to <URL_REG> and insert firstname, lastname, email, username and password	- email or username is already used by another user.		A message is displayed "Email or username is not available".		
2.1	Login with registed user successfully	Go to <URL_LOGIN> insert username and password	user has already registerd		A message is displayed "Login successfully. Welcome back {username}". The dashboard is loaded.		
2.2	Wrong password	Go to <URL_LOGIN> insert username and wrong password			A message is displayed "Wrong password, please try again".		
2.3	Login with empty or wrong username	Go to <URL_LOGIN> insert wrong username			A message is displayed "Invalid username, please try again".		
3.1	Logout successfully	Click Logout button	user has already logged in		A message is displayed "Logout successfully". The login page is reload.		
4.1	Credential recovery	Go to <URL_CREDENTIAL_RECOVER> insert email			A message is displayed "An email with recovery info has been sent to your email". Send an email with credential info to the user email. The login page is reload.		
5.1	Delete account	In Account Management, click "Delete Account"; insert password; click "Yes, I'm sure to delete my account"	user has already logged in		A form for user to choose the reasons why the user want to delete account displayed. The account deleted. The login page is reload.		
6.1	Modify account successfully	In Account Management, click "Modify Account"; insert password; modify the allowed field	user has already logged in		A message is displayed "Modified successfully". The dashboard is reload.		
6.2	Modify account failed	In Account Management, click "Modify Account"; insert password; leave any field empty	user has already logged in		A message is displayed "Modified not successfully". The dashboard is reload.		
7.1	Modify secondary settings successfully	In Setting, modify secondary settings (e.g. theme, layout,...)	user has already logged in		A message is displayed "Modified successfully". The dashboard is reload.		
8.1	Modify AI theming successfully	In AI theming option, inputs a prompt, confirm the modification	user has already logged in		A message is displayed "Modified AI theming successfully". The dashboard is reload.		
9.1	Access profile and statistics	Selects the account display option	user has already logged in		The profile with all its statistics is displayed		
10.1	Change user role	In "User Role" panel, select one of the allowed user roles	user has already logged in		The user role changed following by his dashboard role.		
11.1	Create course	In "Course" panel, select "Create new course", fill the mandatory sections, click "Create new course" final button	user has already logged in, has the right to modify course		Course editor displayed "New course has been created"		
11.2	Modify course	In "Course" panel, select "Modify course", fill the mandatory sections, click "Modify course" final button	user has already logged in, has the right to modify course		Course editor displayed "Course {name} has been modified"		
11.3	Delete course	In "Course" panel, select "Modify course", click "delete course"	user has already logged in, has the right to modify course		Course editor displayed "Course {name} has been deleted"		

N.	Description	Test case data	Precondition	Dep.	Expected result	Actual result	Notes
11.4	Archive course	In "Course" panel, select "Modify course", click "Archive course"	user has already logged in, has the right to modify course		Course editor displayed "Course {name} has been archived"		
11.5	View course	Select course from the dashboard	user has already logged in, has the right to access course		The course is displayed, along with the global ranking if any minigame is present		
11.6	Review course	Select course from the dashboard, leave a review (comment) for the course	user has already logged in, has the right to access course		The review is added to the course		
12.1	Create class	In "Class" panel, select "Create new Class", fill the mandatory sections, click "Create new Class" final button	user has already logged in, has the right to modify Class		Class editor displayed "New Class has been created"		
12.2	Modify class	In "Class" panel, select "Modify Class", fill the mandatory sections, click "Modify Class" final button	user has already logged in, has the right to modify Class		Class editor displayed "Class {name} has been modified"		
12.2	Terminate class	In "Class" panel, select "Modify Class", click "Terminate Class" final button	user has already logged in, has the right to modify Class		Class editor displayed "Class {name} has been terminated"		
12.3	Archive class	In "Class" panel, select "Modify Class", click "Archive Class"	user has already logged in, has the right to modify Class		Class editor displayed "Class {name} has been archived"		
12.4	View class	Selects a class from the dashboard or from an invitation	user has already logged in, has the right to access Class		The public information of the class is displayed		
12.5	Display class' attendees	Selects a class user enrolled into	user has already logged in, has a class enrolled		The list of people attending the class is displayed		
12.5	Display class' statistics	Selects a class user manage	user has already logged in, has the right to modify Class		The performance and statistics of all the attendees is displayed		
12.6	Join class	Selects "Join class" option from dashboard or an invitation	The user is registered, logged in and has the right to join the class. Moreover the class is opened and public		The information of the class is displayed		
12.7	Leave class	Selects "leave class" option from Class panel	The user is registered, logged in and is part of a class		Display message "Leave class successfully". The dashboard is reload.		
13.1	Publish article	In "Article" panel, select "Create new article", fill the mandatory sections, click "Create new article" final button	user has already logged in, has the right to modify article		Aricle editor displayed "New article has been published"		
13.2	Modify article	In "Article" panel, select "Modify Article", fill the mandatory sections, click "Modify Article" final button	user has already logged in, has the right to modify Article		Article editor displayed "Article {name} has been modified"		
13.3	Delete article	In "Article" panel, select "Modify Article", click "delete Article"	user has already logged in, has the right to modify Article		Article editor displayed "Article {name} has been deleted"		
13.4	Archive article	In "Article" panel, select "Modify Article", click "Archive Article"	user has already logged in, has the right to modify Article		Article editor displayed "Article {name} has been archived"		
13.5	View article	Select Article from the dashboard	user has already logged in, has the right to access Article		The Article is displayed		
13.6	Review article	Select Article from the dashboard, leave a review (comment) for the Article	user has already logged in		The review is added to the Article		
14.1	Create minigame	In "Minigame" panel, select "Create new minigame", fill the mandatory sections, click "Create new minigame" final button	user has already logged in, has the right to modify minigame		Minigame editor displayed "New minigame has been created"		

N.	Description	Test case data	Precondition	Dep.	Expected result	Actual result	Notes
14.2	Modify minigame	In "Minigame" panel, select "Modify Minigame", fill the mandatory sections, click "Modify Minigame" final button	user has already logged in, has the right to modify Minigame		Minigame editor displayed "Minigame {name} has been modified"		
14.3	Delete minigame	In "Minigame" panel, select "Modify Minigame", click "delete Minigame"	user has already logged in, has the right to modify Minigame		Minigame editor displayed "Minigame {name} has been deleted"		
14.4	Archive minigame	In "Minigame" panel, select "Modify Minigame", click "Archive Minigame"	user has already logged in, has the right to modify Minigame		Minigame editor displayed "Minigame {name} has been archived"		
14.5	Register minigame from developer	In "Minigame" panel, select "Register minigame to course", select a course, fill mandatory sections, click Accept	user has already logged in, has the right to modify Minigame		Minigame editor displayed "Minigame {name} is register successfully to the course(s)"		
15.1	Pay developer	Select Payment option	user has already logged in and has tasked a developer with a minigame		A message displayed "Payment successfully"		
16.1	Chat with tech support	Select "Chat with supporter"	user has already logged		Chat box is appeared		
17.1	Chat with developer	Select "Chat with developer"	user has already logged		Chat box is appeared		
18.1	Search material	Select Search box, search for a keyword	user has already logged		Dashboard updated with materials user search for		
19.1	Remove review	Remove review from any material	user has already logged, made a review		A message displayed "Review removed successfully"		

7 Git strategy

NOTE: The first part of the actual Git history is not represented here accurately, as we were still refining our approach. This diagram showcases the idealized approach we have adopted and are using moving forward. It combines real elements from our Git history with some fictional commits and branches to illustrate key concepts concisely. Most merges are simplified into single arrows for clarity, and commit messages are omitted for brevity and focus.

7.1 Tweaked Git Flow Strategy Overview

"Tweaked Git Flow" is a customized branching strategy designed to meet our team's specific project management needs. It integrates conventional commits and precise branch naming conventions to ensure clarity and traceability throughout the development and release lifecycle.

7.2 Branch Types in Tweaked Git Flow

1. **Stable Branch (`stable`)**
   - **Purpose**: Represents the stable production-ready state of the codebase.
   - **Workflow**:
     - Reflects the latest stable release.
     - Merges from `develop` for approved updates or critical fixes.
     - Tags releases for deployment.
2. **Hotfix Branches (`hotfix/`)**
   - **Purpose**: Fixes critical production issues.
   - **Naming Convention**: `hotfix/103-login-api-bypass-fix`
   - **Workflow**:
     - Create from `stable`.
     - Merge into `stable` and `develop` after validation.
3. **Release Branches (`release/`)**
   - **Purpose**: Prepares for a new production release.
   - **Naming Convention**: `release/1-D1`, `release/100-0.1.0`
   - **Workflow**:
     - Create from `develop`.
     - Finalize the release (versioning, tagging).
     - Merge into `stable` and `develop` after testing and validation.
4. **Develop Branch (`develop`)**
   - **Purpose**: Integration branch for ongoing development.
   - **Workflow**:
     - Branch off from `main` (or `stable` for initial setup).
     - Merges feature, bugfix, and documentation branches.
     - Prepares releases (`release/` branches).
5. **Documentation Branches (`docs/`)**
   - **Purpose**: Updates or enhances documentation and documents.
   - **Naming Convention**: `docs/22-definition-of-tests`, `docs/23-git-strategy`
   - **Workflow**:

  - Create from `develop`.
  - Merge into `develop` after updates.

6. **Feature Branches (`feat/`)**
   - **Purpose**: Develops new features or significant changes.
   - **Naming Convention**: `feat/101-login-api`
   - **Workflow**:
     - Create from `develop`.
     - Merge into `develop` after completion.

7.3 Conventional Commits in Tweaked Git Flow

Conventional commits provide a structured format for commit messages, facilitating clear communication and automated release notes generation. Each commit message follows the format `<type>: <description>`, where `<type>` indicates the nature of the change. Here are the commit types and their corresponding branch types in our "Tweaked Git Flow" strategy:

- **hotfix**: Fixes critical production issues.
  - Corresponding branch type: `hotfix/103-login-api-bypass-fix`
- **release**: Prepares for a new production release.
  - Corresponding branch types: `release/1-D1`, `release/100-0.1.0`
- **docs**: Updates or enhances documentation.
  - Corresponding branch types: `docs/22-definition-of-tests`, `docs/23-git-strategy`
- **feat**: Used for new feature additions.
  - Corresponding branch type: `feat/101-login-api`
- **fix**: Addresses specific bugs.
  - Corresponding branch type: `fix/102-wrong-email-type-in-login-api`

7.4 Tweaked Git Flow Strategy Overview

NOTE: The first part of the actual Git history is not represented here accurately, as we were still refining our approach. This diagram showcases the idealized approach we have adopted and are using moving forward. It combines real elements from our Git history with some fictional commits and branches to illustrate key concepts concisely. Most merges are simplified into single arrows for clarity, and commit messages are omitted for brevity and focus.

"Tweaked Git Flow" is a customized branching strategy designed to meet our team's specific project management needs. It integrates conventional commits and precise branch naming conventions to ensure clarity and traceability throughout the development and release lifecycle.

7.5 Branch Types in Tweaked Git Flow

1. **Stable Branch (`stable`)**
   - **Purpose**: Represents the stable production-ready state of the codebase.
   - **Workflow**:
     - Reflects the latest stable release.
     - Merges from `develop` for approved updates or critical fixes.
     - Tags releases for deployment.

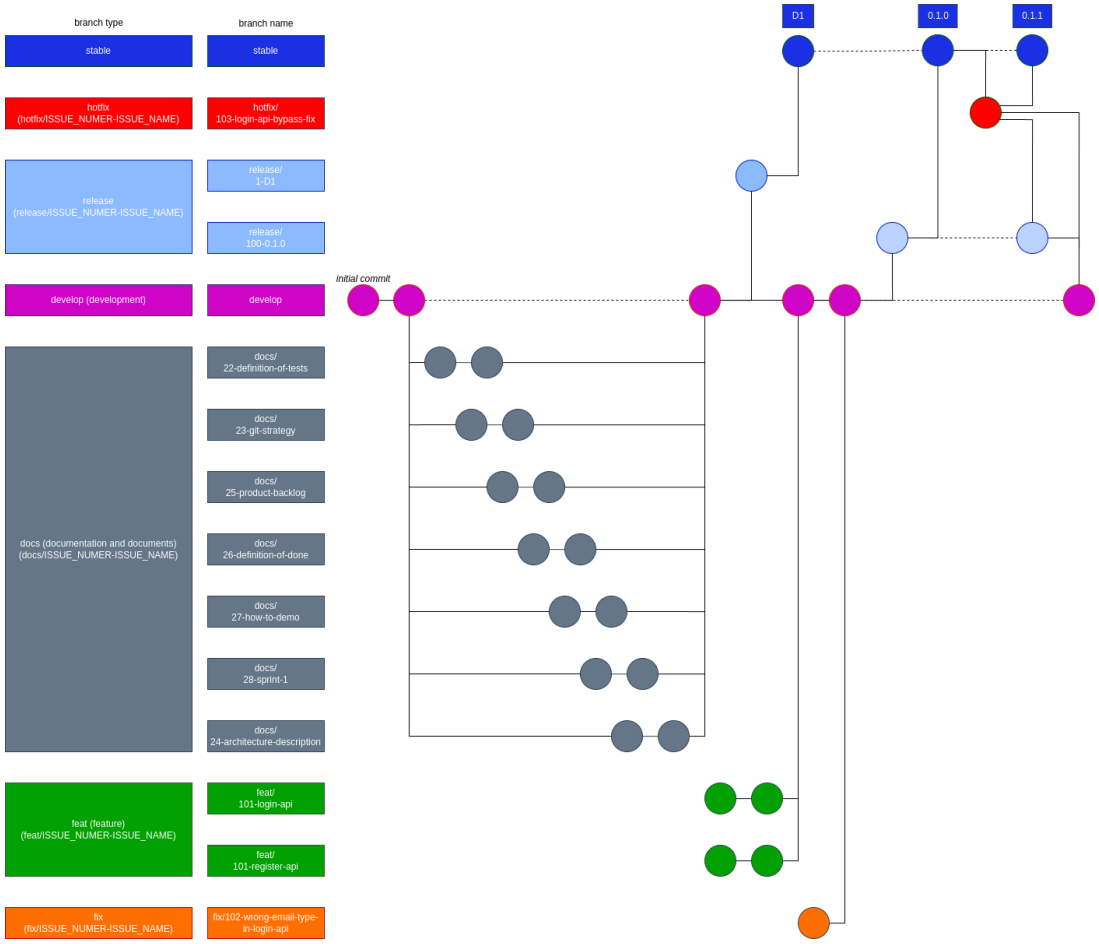

Figure 5: Git strategy example graphic visualization

2. **Hotfix Branches (`hotfix/`)**
   - **Purpose**: Fixes critical production issues.
   - **Naming Convention**: `hotfix/103-login-api-bypass-fix`
   - **Workflow**:
     - Create from `stable`.
     - Merge into `stable` and `develop` after validation.
3. **Release Branches (`release/`)**
   - **Purpose**: Prepares for a new production release.
   - **Naming Convention**: `release/1-D1`, `release/100-0.1.0`
   - **Workflow**:
     - Create from `develop`.
     - Finalize the release (versioning, tagging).
     - Merge into `stable` and `develop` after testing and validation.
4. **Develop Branch (`develop`)**
   - **Purpose**: Integration branch for ongoing development.
   - **Workflow**:
     - Branch off from `main` (or `stable` for initial setup).

  - Merges feature, bugfix, and documentation branches.
  - Prepares releases (`release/` branches).

5. **Documentation Branches (`docs/`)**
   - **Purpose**: Updates or enhances documentation and documents.
   - **Naming Convention**: `docs/22-definition-of-tests`, `docs/23-git-strategy`
   - **Workflow**:
     - Create from `develop`.
     - Merge into `develop` after updates.
6. **Feature Branches (`feat/`)**
   - **Purpose**: Develops new features or significant changes.
   - **Naming Convention**: `feat/101-login-api`
   - **Workflow**:
     - Create from `develop`.
     - Merge into `develop` after completion.

7.6 Conventional Commits in Tweaked Git Flow

Conventional commits provide a structured format for commit messages, facilitating clear communication and automated release notes generation. Each commit message follows the format `<type>: <description>`, where `<type>` indicates the nature of the change. Here are the commit types and their corresponding branch types in our "Tweaked Git Flow" strategy:

- **hotfix**: Fixes critical production issues.
  - Corresponding branch type: `hotfix/103-login-api-bypass-fix`
- **release**: Prepares for a new production release.
  - Corresponding branch types: `release/1-D1`, `release/100-0.1.0`
- **docs**: Updates or enhances documentation.
  - Corresponding branch types: `docs/22-definition-of-tests`, `docs/23-git-strategy`
- **feat**: Used for new feature additions.
  - Corresponding branch type: `feat/101-login-api`
- **fix**: Addresses specific bugs.
  - Corresponding branch type: `fix/102-wrong-email-type-in-login-api`

8 Definition of done

8.1 Overview

The Definition of Done (DoD) is a comprehensive checklist that must be completed for each task, user story, or feature to be considered complete. This section defines the criteria that must be met for a task to be considered "done" for RadiantIQ development.

8.2 Criteria

8.2.1 Code of conduct

During or before code development every member of the team must take into account the following indications.

- Every trance of code should be appropriately commented
- Every logical trance of the code should be documented in order to describe its logic
- Each element of the code should have a meaningful name and follow a coherent notation
- Every commit of a completed task should follow the adequate naming and git policy
- Each modification to existing code should originate from a task, or be a decision taken considering the whole team
- Every addition to the application should be discussed, approved, put in the project backlog and assigned to a sprint before being implemented

8.2.2 Testing

- Unit tests have been written and cover all new and changed functionality.
- All unit tests passed.
- Integration tests have been performed to ensure new changes do not break existing functionality.
- Functional tests have been conducted to verify the feature works as expected.
- Regression tests have been run and passed to ensure existing functionality is not affected.
- Performance tests have been conducted, and performance metrics meet acceptable thresholds.
- No critical bugs remain unresolved.
- All tests in Github Actions are passing.

8.2.3 Documentation

- Code is documented, including inline comments where necessary.
- Public APIs have clear and complete documentation.
- User-facing documentation has been updated (if applicable).
- Release notes have been updated to reflect the new changes.

8.2.4 Security

- Security vulnerabilities have been pointed out.
- Dependencies have been reviewed and updated to ensure there are no security vulnerabilities.
- Data privacy concerns have been addressed.

8.2.5 Deployment and Release

- The feature or fix has been deployed to a staging environment.
- All deployment scripts have been tested and are functioning correctly.
- The feature or fix has been demonstrated to stakeholders and feedback has been incorporated.
- Rollback plans have been prepared in case of deployment failure.
- Code is compatible with at least the top 3 most popular browsers
- Code is compatible with at least one mobile device (android based)

8.3 Process

8.3.1 Review and Approval

- Each task or feature must undergo a code review process.
- Code respects the associated functional requirements completely
- Code runs in accordance with the specified non-functional requirements
- Reviewers must verify that all DoD criteria are met before approving the task.

8.3.2 Quality Assurance

- Quality Assurance (QA) engineers will conduct testing to ensure all criteria are met.
- Any issues identified during testing must be resolved before the task is considered done.

8.3.3 Deployment

- Ensure that the deployment pipeline is green and the feature is deployable.
- Verify that all necessary documentation and release notes are prepared.
- Verify that the feature has been deployed to Render and is functioning as expected.

9 Sprint 1

This section contains the sprint 1 Retrospective, Review and Burndown chart

9.1 Retrospective

We mostly did well, the load assigned was appropriate. We ended up finishing most things (85% of the tasks were completed, and a full depiction of what was done is in the review). As in the end we did not complete any item fully, we should split tasks even more

9.2 Review

All the work done is indicated here, with details about completion:

- Definition of tests 80% done: defined a clear guideline for the tests including objectives, testing types, testing method approaches, testing environment and report tracking. So the developers have a clear vision and write efficient test cases. The actual tests are missing, but they should take little time, having the template
- Environment setup 100% done: setup the backend based on microservices approach. Containerization with docker, CI/CD with Github Action, deployment with Render
- Architecture description 100% done: completed documentation based on the microservices setup
- API definition 30% done: API research completed, now the actual implementation of GraphQL APIs is missing
- Git strategy: 95% done: the proposal model is defined. There are still minor inconsistencies between git flow type branches and commit message types which should be dealt with
- Product backlog: 99% done: The textual file and its translation into gitHub milestones are complete and redundant with various versions. The backlog items for sprint 1 items have also been inserted, but the definition of their syntax is still not totally clear and ust be refined for sprint 2
- Definition of done: 100% done: the textual file has been completed and the criteria for done have been decided in accordance with the course's slides
- Burndown chart automation 70% done: the GitHub part is totally set up, it will be 100% functional for sprint 2

9.3 Burndown chart

This is the burndown chart for sprint 1