Online Lesson Management

System 1. Introduction

1.1 Purpose of the Document

This document aims to provide a detailed description of the "Online Lesson Management System" project. It will specify the system's objectives, functional and non-functional requirements, involved actors, use cases, and project constraints.

1.2 Intended Audience

- **Prof. Orazio Tomarchio:** Software Engineering Professor.
- **Project Team Members:**

Name	**Student code**
Ane C. Kanestrøm	1000084246
David K. Foss	1000072429
Diego Vallespin	1000083350
Henrik L. Jacobsen	1000083298
Vittorio Toscano	1000083356

- **Stakeholders:** Freelancers (tutors) and clients (students).

1.3 System Scope

The system allows freelancers to independently manage online lessons, eliminating third-party commissions and fostering a direct relationship with students.

2. General Description

2.1 System Objectives

- Eliminate intermediary platform commissions.

- Allow professionals to define custom prices and packages.
- Facilitate the management of bookings and cancellations.
- Improve affordability for students.

2.2 Involved Actors

- **Freelancer (Administrator):** Manages lessons, pricing, and interactions.
- **Registered User (Client):** Books lessons and interacts with the professional.
- **Visitor:** Can explore information without registering.

3. System Requirements

3.1 Functional Requirements

- **FR1:** The system must allow user and professional registration and login.
- **FR2:** Professionals must be able to create, modify, and delete lessons.
- **FR3:** Users must be able to book lessons through an interactive calendar.
- **FR4:** The system must handle secure payments for single lessons and packages.
- **FR5:** Users must be able to leave feedback and reviews.
- **FR6:** The system must enable communication
- **FR7:** Professionals must be able to create lesson packages with custom pricing.
- **FR8:** The system must notify users of any changes or cancellations of lessons.

3.2 Non-Functional Requirements

- **NFR1:** The system must ensure personal data protection through advanced encryption.
- **NFR2:** The system's response time must be under 2 seconds even under load.
- **NFR3:** The user interface must be intuitive, responsive, and accessible
- **NFR4:** The system must be scalable to handle an increasing number of users without performance degradation.
- **NFR5:** The system must support daily automatic backups and data recovery mechanisms.

- **NFR6:** Cross-platform compatibility (desktop, mobile, tablet) must be ensured.

4. Use Case Model

4.1 Identified Use Cases

- **UC1:** A user registers as a student or a tutor
- **UC2:** A student books a lesson
- **UC3:** A tutor can manage lesson packages
- **UC4:** A student sends a lesson customisation request
- **UC5:** A tutor can manage his/her lessons and prices
- **UC6:** A tutor or student cancels a lesson
- **UC7:** A student views the lesson calendar
- **UC8:** A student can review tutors after a lesson
- **UC9:** A student can obtain information about a tutor (ratings, background etc.)
- **UC10:** A user can choose a study field to get a list of all relevant tutors
- **UC11:** A student can view the different lessons available for the specific tutor

4.2 Detailed Use Cases

See: “02 - Use Case Model”

5. Implementation and Testing Strategy

5.1 Iterative Development

- The project will be divided into **iterative phases**, with specific objectives for each iteration. Each phase will include the design, development, verification, and validation of incremental features.

5.2 Code Versioning

- Use of **GitHub** for code management.
- Branching strategy: **main** (stable), **develop** (features in progress), **feature/** (specific features).

5.3 Automated Testing

- Use of **JUnit** for testing core features (excluding CRUD operations).
- Testing of key functionalities with documented **Test Cases**.

5.4 Quality Controls

- Regular meetings for code and document reviews.
- Use of **Code Review** and **Continuous Integration (CI)** tools.

6. Glossary

- **UC:** *Use Case*
- **FR:** *Functional Requirement*
- **NFR:** *Non-Functional Requirement*
- **CRUD:** *Create, Read, Update, Delete*
- **UML:** *Unified Modeling Language*
- **JUnit:** Java automated testing tool.
- **CI:** *Continuous Integration*, a methodology for continuous code integration.
- **GitHub:** Hosting platform for version control.
- **Astah:** Tool for creating UML diagrams.