School Master
Analysis and Design Document
Student: Cosma Felicia-Iulia
Group:30431

Revision History

Date	Version	Description	Author
12/04/23	1.0		Cosma Felicia-Iulia

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 5

2.1 Conceptual Architecture 5
2.2 Package Design 5
2.3 Component and Deployment Diagrams 5

III. Elaboration – Iteration 1.2 6

1. Design Model 6

1.1 Dynamic Behavior 6
1.2 Class Design 6

2. Data Model 6

3. Unit Testing 6

IV. Elaboration – Iteration 2 7

1. Architectural Design Refinement 7

2. Design Model Refinement 7

V. Construction and Transition 7

1. System Testing 7

2. Future improvements 7

VI. Bibliography 7

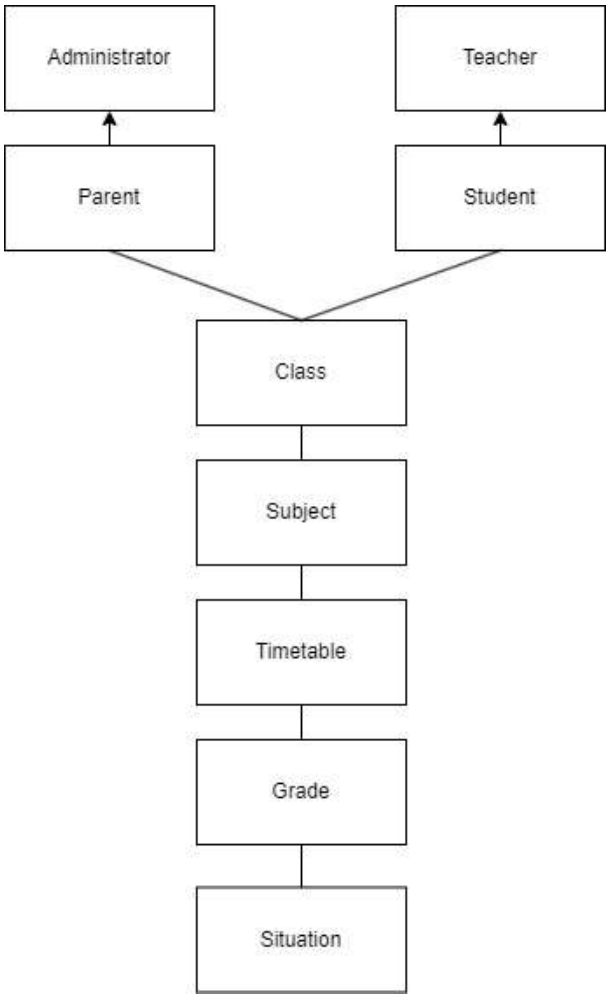
I. Project Specification

Design and implement a client-server application for school or high school administration which can be used by teachers, students, parents, and an administrator. The system should allow teachers to view their timetable, view the school situation of a student, assign grades to students, and finalize the students' situation at the subject they teach. Students should be able to view their timetable and school situation. Parents should be able to view their children's timetable and school situation. The administrator should be able to perform CRUD operations on students, teachers, and classes, assign students to classes, assign teachers to classes, and close the situation of a student at the end of a semester. The system should automatically send an email to a parent when their child receives a grade and when their situation is closed.

II. Elaboration – Iteration 1.1

1. Domain Model

The domain model consists of the following classes: Administrator, Teacher, Student, Parent, Class, Subject, Timetable, Grade, and Situation. The conceptual class diagram is shown below:



2. Architectural Design

2.1 Conceptual Architecture

The system will be designed using a layered architecture, which is a common architectural style for client-server applications. This architecture separates the system into layers, with each layer having a specific responsibility and interacting only with the layer directly below or above it.

The layers of the system are as follows:

1. Presentation Layer: This layer is responsible for presenting information to the users and receiving input from them. It will be implemented using a web-based user interface, which will allow users to access the system from any device with an internet connection.
2. Application Layer: This layer will contain the business logic of the system, including the processing of requests from the users, the validation of input data, and the execution of database queries. It will be implemented using a RESTful API, which will allow the presentation layer to communicate with the application layer using HTTP requests and responses.
3. Persistence Layer: This layer will be responsible for storing and retrieving data from the database. It will be implemented using a relational database management system (RDBMS), which will provide a structured and scalable storage solution for the system.

By using a layered architecture, we can ensure that the system is modular, scalable, and maintainable. Each layer can be developed and tested independently, allowing for easier debugging and refactoring. Additionally, the separation of concerns makes it easier to implement changes to one layer without affecting the others.

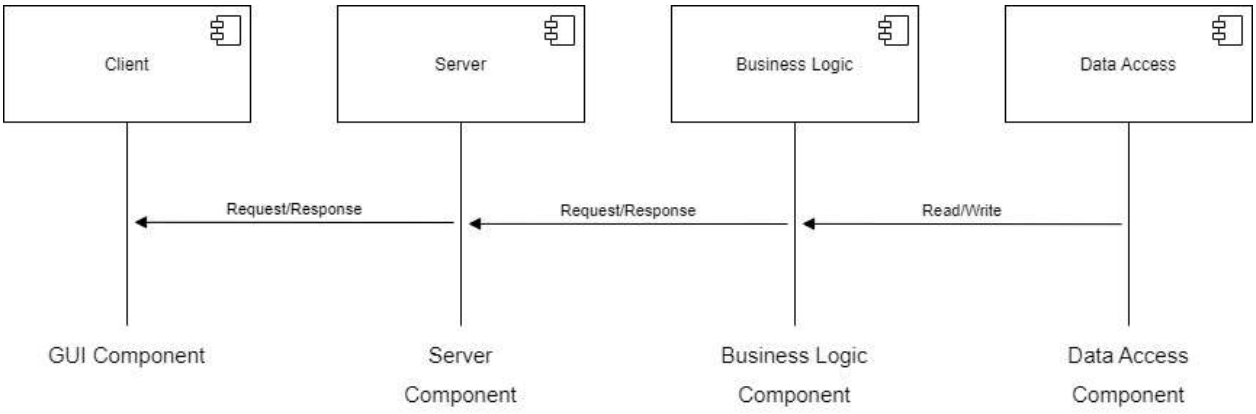
2.2 Package Design

The package diagram is shown below:

2.3 Component and Deployment Diagrams

The diagram shows the four components of the system: the Client, Server, Business Logic, and Data Access components. The Client component represents the graphical user interface used by teachers, students, and parents, which can also be accessed through a command-line interface by students. The Server component is responsible for handling requests from the clients and coordinating the business logic and data access components. The Business Logic component is responsible for implementing the rules and logic of the system, such as verifying user credentials and managing the assignment of grades to students. The Data Access component is responsible for managing the communication with the database, retrieving and storing data.

The GUI and Business Logic components communicate with each other through well-defined interfaces, as do the Business Logic and Data Access components. The communication between components is shown by the Request/Response and Read/Write arrows in the diagram. Finally, the deployment diagram will show how these components will be deployed on different servers or machines.

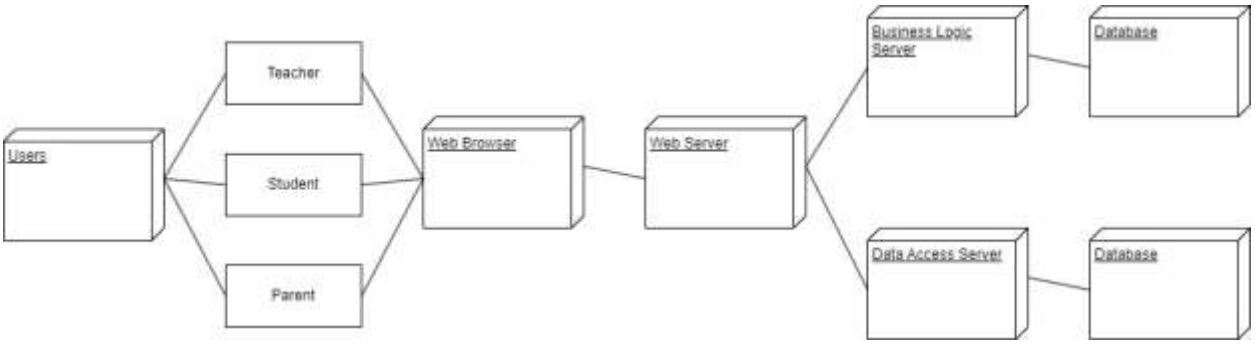


In this deployment diagram, the users access the system through a web browser. The web server component is responsible for serving the web pages to the users, and for handling their requests.

The business logic server component is responsible for implementing the system's business logic, and communicates with the data access server component to retrieve and store data in the databases.

The databases are represented as separate components, with one database for storing information related to users, classes, and timetables, and another database for storing grades and other academic records.

Overall, this deployment diagram shows how the various components of the system will be deployed on different servers or machines, and how they will communicate with each other to provide the necessary functionality to users.



III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

1.2 Class Design

[Create the UML class diagram; apply GoF patterns and motivate your choice]

2. Data Model

[Create the data model for the system.]

3. Unit Testing

[Present the used testing methods and the associated test case scenarios.]

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

VI. Bibliography