PepeFit
Architecture Notebook

1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

2. Architectural goals and philosophy

While we are design our project's architecture, our main goal is simplicity. We don't want to design a complex structure because deployment is way ease with a simple architecture. And our project is not that big to be too complex. There is benefit to say if architecture will be too complex, it can effect our schedule, our application performance and us. Because it may be hard to accomplish. And it does not need to be robust for long-term maintenance.

3. Assumptions and dependencies

- User can want a reliable system.
- User wish to do his/her job fast.
- Storage must be big enough to store all the data.
- UI will be simple and easy to navigate.

Dependencies

- System must be run on a web browser. So we have to design our model and controllers part.
- Data will live in a RDBMS so, we have to design our model components with respect to that.

4. Architecturally significant requirements

- System must be on an RDMBS.
- Interface must be smooth and simple.
- Project must be on an OOP programming language.
- System must be easy to maintenance.

5. Decisions, constraints, and justifications

- System must be respond within 4 seconds.
- System must be easy to use.
- System must be run on a browser.
- Doing Maintenance every month.
- System should be up-to-date to latest technology.
- System shouldn't crash.

6. Architectural Mechanisms

Reliability Provider

System must be reliable all the time whatever happens. So while we are design our architecture, we have to think reliability factor. It's in the design factor.

Error Handler

There must be an error handler because we have to handle the error when it occurs and we have to do it fast. It's in the implementation state.

Recovery Manager

We don't want to lose our whole system while designing or implementing. So it can be good if we have a recovery manager. It's in design and implementation states.

Refinement

While we are design, we don't want to design a complex one. We should check our design and do refinement for every step. It's in the design state.

Security Provider

System must be secure. It's in the design and implementation states.

7. Key abstractions

- Person Class: It's an abstraction for members and trainers.
- Courses-Trainer Relationship: It's an abstraction for schedule.
- Courses Class: It's an abstraction for courses.

8. Layers or architectural framework

PepeFit will use Model-View-Controller Pattern that is a 3 layered model.

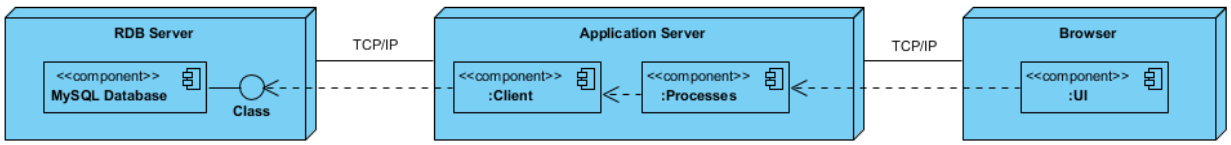
- **Model:** It's an abstraction for a database. All information about trainers, users, courses etc. will store via this layer. And other layer that needs informations will take data via this layer.
- **View:** It's our UI layer. It provides interactions that connect into controller layer.
- **Controller:** It's the middle layer. It provides a connection between model and view.

9. Architectural views

You can see our 4+1 architectural view model.

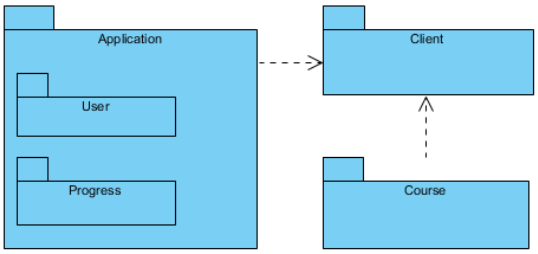
- **Physical View:**

  Deployment Diagram:



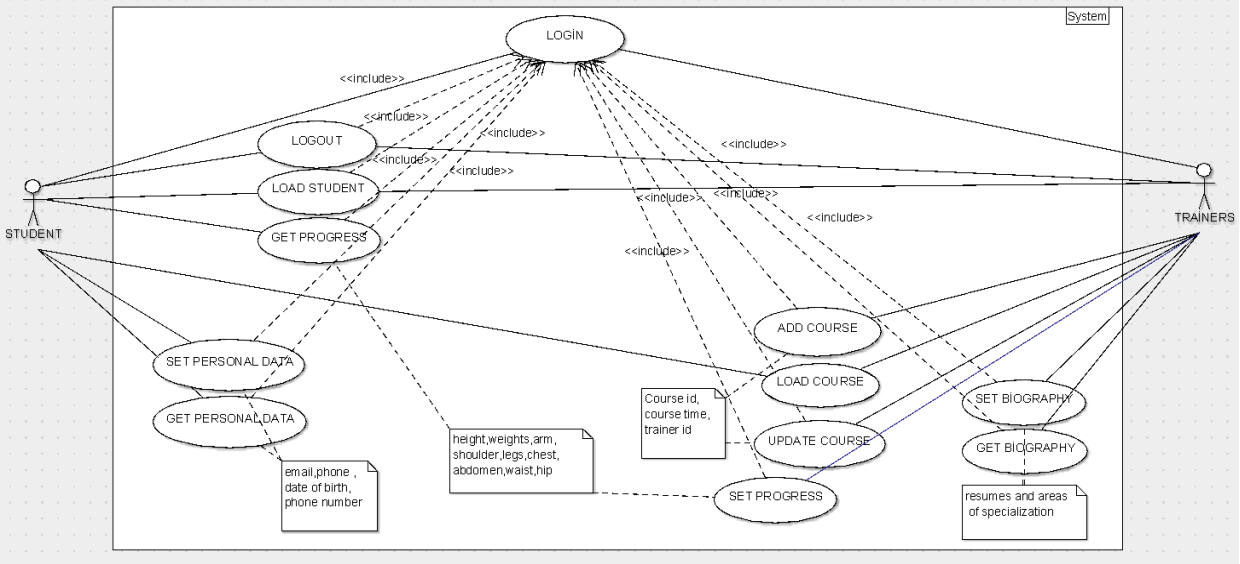
- **Development View:**

  Package Diagram



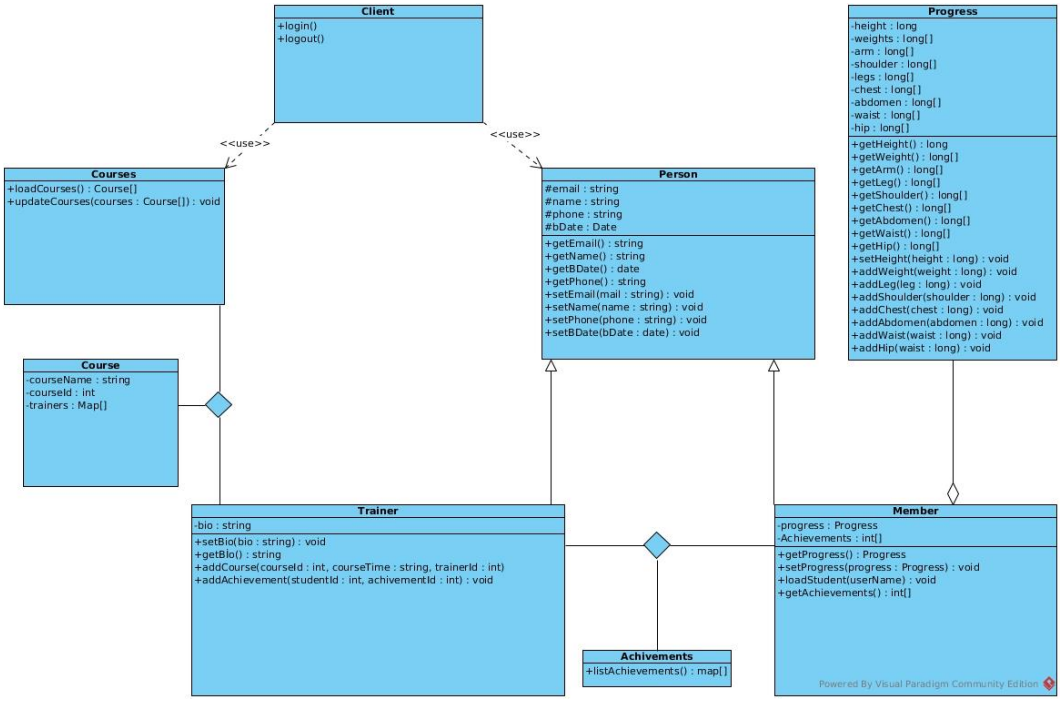
- **Scenario View:**

  Use Case Diagram



- **Logical View:**

  Class Diagram



- **Process View:** You can see Tabular Descriptions of Use Cases on #Del2.