1. **Architecture and design**

This project describes in the terms of object-oriented programming the hierarchical structure of a university. It defines 6 classes – Lecturer, FacultyGuard, Employee, Speciality, Faculty and University.

* 1. **Lecturer class** – it consists of name, address birth year, experience and academic rank. It is a derived class of the Employee class. The salary of the lecturer can be increased annually. He can change his academic rank.
  2. **FacultyGuard class** – it consists of name, address, birth year, experience and eligibility status for having a weapon. It is a derived class of the Employee class. We can add lecturer, faculty guard or speciality to the faculty and can also remove them.
  3. **Employee class** – it consists of name, address, birth year and experience. It is a base class of the Lecturer and FacultyGuard classes and also is an abstract class because it has the pure virtual function *increaseAnnualSalary()* which is overridden in the derived classes in a different way. We can calculate the age of an employee, set its name, address and experience.
  4. **Speciality class** – it consists of name, maximal enrolled people, number of disciplines and disciplines. We can add a discipline and also remove it. Name of the speciality, maximal number of enrolled students and disciplines can be set.
  5. **Faculty class** – it consists of name, number of lecturers, number of faculty guards, number of specialties and lecturers, faculty guards and specialties themselves. We can add or remove a lecturer, a faculty guard or a speciality. Lecturers, faculty guards and specialities can be set. Also the name of the faculty can be changed.
  6. **University class** – it consists of name, year of foundation, name of rector, number of faculties in university, maximal faculties possible in university and the faculties themselves. We can add or remove a faculty, set the name of the faculty and set the name of the rector.

1. **Project structure**

The project has a simple structure:

* Directory named *include* which contains all the header files of the project.
* Directory named *src* which contains all the source files of the project.
* Directory named *test* which contains all the unit test files of the project.
* Directory named *doxygen* which contains the auto generated online documentation browser (in HTML).
* Document named *Documentation* which contains the external documentation of the project
* Source file named *main.cpp* which is the starting point of the project.
* Autogenerated file named *University.cbp* which contains setup for the build.

1. **Future plans for developing the architecture and design of the project and improving its functionality**

We can further develop this project by upgrading its architecture and design. We can create a class representing speciality in the faculty which class will replace the current representation of the specialities in faculty. We could also add at least one more functionality to the lecturer class – it will be possible for the lecturer to set grades. Here comes the abstraction for a student. We will create a class representing student. A student will have name, course, faculty number and array of grades for instance. Grades will be represented with enum type instead of integer type for a better encapsulation of the data. A student will be able to enroll in elective disciplines in the faculty. Another architectural idea is to create a class which represents a campus. It will consists of rooms and because of that we will create a class representing a room. There are also other ways for developing this project, but these concepts are one of the most important in the future development. As a conclusion we can summarize that in the future the project can be developed by adding the key concepts for speciality, student, campus, room etc. By applying these architectural concepts and design solutions we can improve the functionality of the project.