Institute Automation System

System Design

Version 1.0

2 May 2018

Oğuzhan Ulusoy

Prepared for
CSE490 Thesis Project

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

SYSTEM DESIGN DOCUMENT[1]

1. Introduction

1.1. Purpose of the System

Institute Automation System is a web-based application that provides campus interface to academic staffs, institute staffs and grad students. The main goal of system is to satisfy easier and faster operations.

1.2. Design Goals

Institute Automation System suggests more usable, reliable, performance, supportable, secure platform. The target is to implement all offered functionalities in Requirement Analysis Document, in shortly RAD. It is aimed to succeed no delay of database complexity. It is targeted quick operations and easier, more secure, more powerful a system.

1.3. Definitions, Acronyms, and Abbreviations

- IAS: Institute Automation System
- ORM: Object relational model
- RAD: Requirement Analysis Document
- SQL: Standard query language
- VPS: Virtual private server

1.4. References

- Campus Online
- Requirement Analysis Document (version II) of Institute Automation System

2. Current Software Architecture

There is an existing system that has same purposes. The previous system has been implemented approximately fifteen years ago. Existing system, that is called Campus Online, provides fundamental staff operations and students operations, such as to prepare weekly schedule, to display curriculum and so on. It was developed by Asp-Net. Existing system has more complicated database infrastructure. More relations among different tables cause slower operations, sometimes it can cause to deadlock. The system was developed according to smaller population. Web-traffic is not enough. Moreover, there are needed more functionalities. [2]

3. Proposed Software Architecture

Essential requirements are to develop the features of existing system more efficiently. Optional requirements, in other words desired features, are going to be implemented. We are going to develop the system with Django that is a framework that uses Python infrastructure. Because of Django-structure, we can handle the system in shorter time as more useable and useful. We are going to try to decrease the relations among different entities. We are going to

develop the system with different design-pattern to make the system more efficient. We are going to use data-structure to develop the network side of project.

3.1. Overview

We are going to use Django that is a web-framework that uses Python infrastructure. Because of Django-structure, we are going to use model-view-template design pattern that is another type of model-view-controller design pattern. We are going to develop service-based solutions, that is called service-oriented architecture. The complete system is divided into smaller sub-systems with own activities.

3.2. System Decomposition

Institute Automation System is made of four major actors that are named grad students, staffs, system administrators and visitor. Staffs are divided into academic staffs and institute staffs. Each actor in the system has different authorities. Operations are divided into actor authorities. We have four actors to implement, so we have four different sub-systems. We are going to develop institute staff side.

Institute staff is able to manage institute, academic staffs, institute staffs, grad students and applications. Therefore, institute staff module, in other words sub-system, is divided into five sub-system, another level.

- Institute sub-system includes management of institutes, departments, programs, curriculums, courses, course types and sections.
    - Institutes tab under institute sub-system contains to display all data and add a new institute with own attributes.
    - Departments tab under institute sub-system contains to display all data and add a new department with own attributes.
    - Programs tab under institute sub-system contains to display all data and add a new program with own attributes.
    - Curriculums tab under institute sub-system contains to display all data and add a new curriculum with own attributes.
    - Courses tab under institute sub-system contains to display all data, display available courses, define a new course with own attributes, open or close an existing course and remove an existing course.
    - Course types tab under institute sub-system contains to display all data and add a new course type with own attributes.
    - Sections tab under institute sub-system contains to display all data, add a new section with own attributes and remove all data.
- Academic staff sub-system includes management of all academic staffs, institute heads, department heads, program heads and quota managers.
    - All academic staffs tab under academic staff sub-system contains to display all data and add new academic staff with own attributes.
    - Institute heads tab under academic staff sub-system contains to display all data.

Institute Automation System

  - Department heads tab under academic staff sub-system contains to display all data.
  - Program heads tab under academic staff sub-system contains to display all data.
  - Quota managers tab under academic staff sub-system contains to display all data and add new quota manager with own attributes.
- Institute staff sub-system includes management of all institute staffs.
  - All institute staffs tab under institute staff sub-system contains to display all data and provide acceptance operations.
- Grad student sub-system include management of all grad students.
  - All grad students tab under grad student sub-system contains to display all data.
  - All completed courses tab under grad student sub-system contains to display all data about completed courses by department, program, courses, students.
- Application sub-system includes management of all applications from visitors.
  - Applications tab under application sub-system contains to display all data and accept, reject or remove operations.
  - Remove all applications tab under application sub-system contains to remove all applications from system forever.

3.3. Hardware-Software Mapping

We are able to examine hardware-software mapping by two sides that are actors and system.

- In terms of actors, there is no any constraint on hardware-software mapping. Institute Automation System is a web-based application that developed by Python infrastructure. All different type actor is able to reach to the system with ordinary computer, or mobile devices with internet-connection.
- In terms of system, we are going to demonstrate the system with Heroku Cloud technology. Full version of Institute Automation System might able to demonstrate with Digital Ocean, Google or Amazon Cloud services, and so on.

The important point in live version is to run the system with virtual environment and Python infrastructure.

3.4. Persistent Data Management

We are going to work with SQLite3 database. It comes as default database system in Python. SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects. [3]

We are going to work with object relational model, in shortly ORM. Object relational model is to more power, greater flexibility, better performance and greater data integrity then

those that came before it. Some of the benefits that are offered by the Object-Relational Model include:

- Extensibility – We are able to extend the capability of the database server; this can be done by defining new data types, as well as user-defined patterns. This allows the us to store and manage data.
- Complex types – It allows us to define new data types that combine one or more of the currently existing data types. Complex types aid in better flexibility in organizing the data on a structure made up of columns and tables.
- Inheritance – We are able to define objects or types and tables that procure the properties of other objects, as well as add new properties that specific to the object that been defined.

Description of the encapsulation of the database:

We are going to work with Django user. We will use it as abstract user, then other actors are inherited by this abstract user. Grad student, staff and visitor are inherited by abstract user. Academic staff is inherited by staff. Quota manager is inherited by academic staff. Inheritance is provided with foreign key feature of Django model.

- User model - User model needs e-mail, first name, last name, password, phone number.
- Visitor model – Visitor model needs user as foreign key, citizenship number as char field, birthday as date field, gender as char field, application date as date field, address as text field, city as char field, degree as char field, university as char field, general point average as decimal field, ales as positive integer field, yds as positive integer field, acceptance as boolean field, program as foreign key.
- Student model: Student needs user as foreign key, student identification number as char field, student e-mail as e-mail field, curriculum as foreign key, program as foreign key, advisor as foreign key and check variables as boolean fields.
- Staff model: Staff model needs user as foreign key, citizenship number as char field, birthday as date field, gender as char field, main e-mail as e-mail field, school mail as e-mail field, address as text field and city as char field.
- Academic Staff model: Academic Staff model needs staff as one-to-one field from Staff model, university as car field and institute as foreign key.
- Institute model: Institute model needs name as char field, head as foreign key and established date as date field.
- Department model: Department model needs a name as char field, institute as foreign key and head as foreign key.
- Program model: Program model needs name as char field, code as char field, type as char field, thesis as boolean field, department as foreign key, head as foreign key, quota manager as foreign key.
- Curriculum model: Curriculum model needs program as foreign key, year as integer field.
- Course model: Course model needs code as char field, title as char field, description as text field, credit as integer field, ects credit as integer field, program as foreign key,

university as char field and check variables as boolean fields. Course model acts a archive for all courses from beginning to now.

- Quota Manager model: Quota Manager model needs quota manager as foreign key. It is inherited from academic staff model.
- Section model: Section model needs course as foreign key, number as positive integer field, quota as positive integer field, year as char field, instructor as foreign key, semester as char field, special quota as many-to-many field, students as many-to-many field. Section is open course.
- Schedule model: Schedule model needs section as foreign key, day as char field, slot as char field and place as char field. Instances of Schedule model are to show where and when a section is.
- Course Type model: Course Type model needs title as char field and code as char field.
- CCR Course model: CCR Course model needs curriculum as foreign key, type as foreign key, no as positive integer field and semester as positive integer field. This model is to complete all slots with courses in curriculum.
- Offered Course model: Offered Course model needs ccr course as foreign key, active course as a foreign key. Ccr course is inherited from ccr course model and active course is inherited by course model.
- Taken Course model: Taken Course model needs student as foreign key, ccr course as foreign key, active course as foreign key and acceptance as null boolean field. Ccr course is a course that has defined according to curriculum. Active course is a course that is inherited from Section, student desire to take. Acceptance is statue of the data that is required why advisor permission. Uniqueness is checked.
- Completed Course model – Completed Course model needs student as foreign key, ccr course as foreign key, active course as foreign key and grade as char field. Ccr course is a course that has defined according to curriculum. Active course is a course that student has passed. Grade is a semester point of the active course. Uniqueness is checked.

All uniqueness check is controlled in models. All lengths are set up in models. Proxy design-pattern is used to prevent to models directly.

3.5. Access Control and Security

All defined users on Institute Automation System, those are academic staffs, institute staffs and grad students, are able to login in to the system with school e-mail that is defined in registration process and own password in secure.

System administrator that is a user has all management authentication, can reach in to the control management mechanism of Institute Automation System over Django administrator panel with own username and password.

Visitors are able to reach to application forms to become grad student directly without any authentication mechanism.

All permissions are defined as groups on Django administrator panel by system administrator. These groups are set up through actors.

Actor name	Group name
Academic Staff	Academic Staffs
Institute Staff	Institute Staffs
Grad Student	Grad Students

Staffs can be divided into another level in later with same logic. After division of groups, operations have been defined. Accordingly, institute staff can make adding, displaying or deleting operations anymore. Thus, because of front-end structure, anybody cannot reach another actor's permissions.

3.6. Global Software Control

There is required internet-connection that was established earlier, to reach to Institute Automation System. Because of HTTP messages, people can access to Institute Automation System. Session is detected on authentication. After authentication, person is redirected in to another page. Her group is detected, so that she cannot access to page that she does not have permission to display.

Accessible pages are synchronized over internet-connection. To succeed this, database connection is also required. After database connection has been done, synchronization operations are satisfied.

3.7. Boundary Conditions

- Start-up:
  - There is required a virtual private server, in shortly VPS, that has Python 3 version. Then, Django 2 version must be set up.
  - Project's requirements must be documented with freeze command. It creates a text file that is named requirements.
  - Debug feature in project settings must be converted from true to false.
  - Project must be transferred from local repository in to this virtual private server completely.
  - Created requirements text file must be called with command line. Thus, other requirements are downloaded from relevant libraries.
  - The server must be run finally.

- Shutdown:
  - Shutdown might occur with two-ways.
    I. Physical hardware might be interrupted.
    II. Virtual private server might be stopped.

- Error behavior:
  - Internet-connection might be unconnected.
  - Physical hardware might be interrupted.
  - Virtual private serve might be stopped.
  - Database might be lost.
  - Database connection might be unconnected.

4. Subsystem Services

Subsystem services will be detailed in Object Design Document. We are going to introduce information in System Decomposition part.

- Institute sub-system includes management of institutes, departments, programs, curriculums, courses, course types and sections.
    - Institutes tab under institute sub-system contains to display all data and add a new institute with own attributes. Adding operation creates institute object, then saves into Institute model.
    - Departments tab under institute sub-system contains to display all data and add a new department with own attributes. Adding operation creates department object, then saves into Department model.
    - Programs tab under institute sub-system contains to display all data and add a new program with own attributes. Adding operation creates program object, then saves into Program model.
    - Curriculums tab under institute sub-system contains to display all data and add a new curriculum with own attributes. Adding operation creates curriculum object, then saves into Curriculum model.
    - Courses tab under institute sub-system contains to display all data, display available courses, define a new course with own attributes, open or close an existing course and remove an existing course. Adding operation creates course object, then saves into Course model. Removing operation changes a check variable from false to true. Open/close operation changes a check variable from false to true bidirectionally.
    - Course types tab under institute sub-system contains to display all data and add a new course type with own attributes. Adding operation creates course type object, then saves into Course Type model.
    - Sections tab under institute sub-system contains to display all data, add a new section with own attributes and remove all data. Adding operation creates section object, then saves into Section model. Removing operation removes from database forever.
- Academic staff sub-system includes management of all academic staffs, institute heads, department heads, program heads and quota managers.
    - All academic staffs tab under academic staff sub-system contains to display all data and add new academic staff with own attributes. Adding operation creates academic staff object, then saves into Academic Staff model.
    - Institute heads tab under academic staff sub-system contains to display all data.
    - Department heads tab under academic staff sub-system contains to display all data.
    - Program heads tab under academic staff sub-system contains to display all data.

    - Quota managers tab under academic staff sub-system contains to display all data and add new quota manager with own attributes. Adding operation creates quota manager object, then saves into Quota Manager model.
- Institute staff sub-system includes management of all institute staffs.
    - All institute staffs tab under institute staff sub-system contains to display all data and provide acceptance operations.
- Grad student sub-system include management of all grad students.
    - All grad students tab under grad student sub-system contains to display all data.
    - All completed courses tab under grad student sub-system contains to display all data about completed courses by department, program, courses, students.
- Application sub-system includes management of all applications from visitors.
    - Applications tab under application sub-system contains to display all data and accept, reject or remove operations. Accepting operation creates student object, then saves into Student model. Removing operation removes the selected instance from database forever.
    - Remove all applications tab under application sub-system contains to remove all applications from system forever.

5. References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. Campus Online, www.campus.isikun.edu.tr
3. SQLite Definition from formal web-site, www.sqlite.org/about.html