

**Clubs & Events**

Object Design

Version 1.0

**Begüm Bilgin**

**Murat Akalın**

**Mert Çetinkaya**

**S. Murat Yiğit**

Prepared for

CSE490 Project



Table of Contents

[**1.** **Introduction** 2](#_Toc535406844)

[**1.1** **Object Design Trade-offs** 2](#_Toc535406845)

[**1.2** **Interface Documentation Guidelines** 3](#_Toc535406846)

[**1.3** **Definitions, Acronyms and Abbreviations** 3](#_Toc535406847)

[**1.4** **References** 3](#_Toc535406848)

[**2.** **Packages** 3](#_Toc535406849)

[**3.** **Class Interfaces** 4](#_Toc535406850)

[**4.** **References** 7](#_Toc535406851)

**OBJECT DESIGN DOCUMENT [1]**

# **Introduction**

## **Object Design Trade-offs**

* Instead of using a Database-First approach, we've decided to use code-first approach in the entity framework due to the fact it provides auto-generated code, reducing development time.
* We are using Django Framework, it uses default SQLITE3 database. This database can change but we have not changed it because, it meets our needs in the system. Django Authentication system will work perfect with SQLITE3. All login processes and registration processes will automatically save by Django. This authentication includes password encryption, password check, and user name check. It means this user name used before or not. For the same thing for the user e-mail. Authentication will check this e-mail used in before in the system. Also, authentication will check password length and password content which is your password must contain at least 8 characters, your password may not be very similar to your other personal information, your password cannot be a used password and your password cannot be fully numeric. All these things will check with Django authentication. But this authentication stills not enough for us. So we used csrf token in the HTML codes. It is a kind of encryption. Also we wrote some security codes on the views. This codes will run before the calling the views. It means when you open a page, first attached to view will run. Because view will call page which is the you want. Our security code will first run. If user hasn’t authority for that page, system will logout the user and returns the home page.
* Our system has four actors which are visitors, registered users, moderator and admin. Except the visitors, we are using just one table for the all users. All users have mostly same attributes which are first name, last name, e-mail, and password. We are keeping these attributes on same table. We are using Django Admin page for the for authorization operations. So our database complexity will very small. But we will add some admin functions on our web-site moderator page. Because Mr. Onur wanted it.
* For the Clubs and Events, we added ‘published’ and ‘waiting’. Because, if moderator want to delete any clubs or events, he/she cannot. So he/she can disable them. These attributes are Boolean attributes actually.
* While we have invested development time for UI to look appealing, our main focus was for the system functional requirements to work properly. But we changed something on home page; we added timeline for the user favorite clubs and events. If user is not login the system, visitors will see calendar on the right side. If user login the system, and also if user has favorite clubs on the system, these clubs’ events will represent home page timeline on the right side. As a result, our UI’s look simple but easy to use. All of our functional requirements are satisfied more than RAD document.

## **Interface Documentation Guidelines**

Front-end design:

* Static and dynamic substance, for example, CSS and Bootstrap files, contents and pictures are kept in a document called content.
* Django Framework will use SQLITE3 database, it’s simple and basic. We connected database between the views and url's. But main thing is our models; it is automatically create tables on database. This system using for building dynamic.
* Every performing actor in the framework has their predefined format page where resources are held. This format is the equivalent for all pages of an on-screen character, with content or body area changing accordingly.
* Bootstrap 3.3.7 is used for styling user interface elements.HTML forms styled with bootstrap are used for input operations. They use HTTP GET/POST, and secured by validating anti forgery token which is csrf token, in controllers and views.
* Django Admin Panel is used for Administration (delete processes, staff processes. Etc.)
  1. **Definitions, Acronyms and Abbreviations**
* RAD: Requirements Analysis Document
* SDD: System Design Document
* Clubs & Events: Activity Page of Işık University
* View is a visual representation of a model.
* DB is short version of database term.

## **References**

* RAD of Clubs & Events
* SDD of Clubs & Events
* Model Implementation

# **Packages**

* Clubs&Events Project
* .vs
  + Clubs&Events
  + slnx.sqlite
  + VSWorkspaceState.json
* docs
* MainProject
  + media
  + migrations
    - 0001\_initial.py
    - 0002\_auto\_20190429\_2009.py
    - \_init\_.py
  + templates
    - images
    - add\_comment.html
    - allcomments.html
    - allcomments\_detail.html
    - anasayfa.html
    - changeUserStatus.html
    - edit\_club.html
    - edit\_club\_detail.html
    - edit\_event.html
    - edit\_event\_detail.html
    - favorite\_clubs.html
    - KulupDetay.html
    - kulupler.html
    - Login.html
    - ModeratorPanel.html
    - new\_club.html
    - new\_event.html
    - Register.html
  + \_Init.py\_
  + admin.py
  + apps.py
  + forms.py
  + models.py
  + tests.py
  + urls.py
  + views.py
* MuratProject
  + \_init\_.py
  + settings.py
  + urls.py
  + wsgi.py
* venv
  + Lib
  + Sctipts
* db.sqlite3
* manage.py

# **Class Interfaces**

Model Name: User (Registered User, it has no status on the system.)

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* First name – First name of the user.
* Last name – Last name of the user.
* Username – Unique username of the user.
* E-mail – E-mail address of the user.
* Id – Unique user id.
* Password - Password of the user.
* Date joined – Date that user has been joined into the system.
* Is active – It is about instant user activity
* Is super user – It is a status for defining system administrator.
* Is staff – It is a status for defining staff. (Moderator)

*Dependencies*:

* None.

Model Operations:

* There is no model operation.

Model Name: Admin

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* All user information of Admin comes from the User model.
* Admin has just status which is super user, with this status; he/she can administer the system.

*Dependencies*:

* None.

Model Operations:

* There is no model operation.

Model Name: Moderator

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* All user information of Moderator comes from the User model.
* Moderator has just status which is staff, with this status; he/she can login admin panel which is for the moderator side, also he/she can login the moderator page of the web-site.

*Dependencies*:

* None.

Model Operations:

* There is no model operation.

Model Name: Visitor

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

*From Django Official Documentation*:

**django.contrib.auth.models.AnonymousUser** is a class that implements the **django.contrib.auth.models.User** interface, with these differences:

* id is always None.
* username is always the empty string.
* get\_username() always returns the empty string.
* is\_anonymous is True instead of False.
* is\_authenticated is False instead of True.
* is\_staff and is\_superuser are always False.
* is\_active is always False.
* groups and user\_permissions are always empty.
* set\_password(), check\_password(), save() and delete() raise NotImplementedError.

*Dependencies*:

* **django.contrib.auth.models.AnonymousUser** is a class that implements the **django.contrib.auth.models.User** interface.
* Basically, it is dependent to User model.

Model Operations:

* There is no model operation.

Model Name: Events

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* Club – Foreign Key of Club class.
* Event Name – The name of the event.
* Event Created Day – Date that the event is created.
* Event Day – The day of the week that this event will take place.
* Event Location – The location that the event will take place.
* Event Info – Information about the event.
* Status – Status of the event. Shows if the event is Enabled/Active or Disabled/Inactive.

*Dependencies*:

* It is dependent with Clubs model.

Model Operations:

* There is no model operation.

Model Name: Clubs

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* Club Name – The name of the club.
* Club Info – Information given about the club.
* Club Emblem – The logo or the emblem of a club given as an image.
* Status – Status of the club. Shows if the club is Enabled/Active or Disabled/Inactive.
* User Favorite – A list of favorite clubs of the User. (many to many field so it uses foreign key inside. Foreign Key of User class.

*Dependencies*:

* None.

Model Operations:

* There is no model operation.

Model Name: Comments

Model attributes – explanation – dependencies:

*Model attributes – explanation*:

* Event – Foreign Key of Event class.
* User – Foreign Key of User class.
* Comment – A comment text in string written by a User, for an event of a club.
* Status – Indicates the approval status of a comment. Comment has three states which are accepted, waiting and rejected. Comments are waiting for approval by default. A comment can be accepted or rejected. Approvals are given by Admin or Moderator.

*Dependencies*:

* It is dependent with Events and User models.

Model Operations:

* There is no model operation.

# **References**

1. Bruegge B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.
2. <https://sksfit.com>
3. <https://www.djangoproject.com>
4. https://www.python.org