When The Ring Bells

Object Design

1.0

24.12.2019

Ersin ÇEBİ
Zafer KALYONCU
Serdar ŞAHİN
Çağatay DEMİRCAN

Prepared for
SE301 Software Engineering

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

OBJECT DESIGN DOCUMENT

1. Introduction

While designing the application to the users to try to make the fastest way to design, we have stayed between the stable and beautiful design. Database to repeat the data in a short time with the data to keep the data in some data, although we kept some database. In our application integrated with Firebase, we transformed our code structures into a firebase format.

1.1. Object Design Trade-offs

Reliability

We have designed the software with firebase for users can be use the application usually and safely. We have tested all the codes we have written. After each update or any changing, we have added to our project with testing. We have seen and corrected our mistakes easily because we are progressing regularly.

Expandability

While designing our system, we paid attention to extensible. Thanks to its modular structure, our system is ready for new technologies that can be added later.

Programmability

In our project, we have created a simple and functional application using with Angular CLI and firebase. The reason we use the Firebase is that simple and safety. In addition, Angular CLI and Firebase both supported by google. In this way, we have designed the database part very easily and we have written our code according to these designs.

Maintainability

When we completed our project, we provide the necessary technical support. Users can reach our application on every device that has internet connection via web browser. We designed our system under stable and make easy to use.

Compatibility

We have choice to use Angular CLI as typescript framework, because we wanna use a new technology and well structured framework. So, we have decided to design it in Angular CLI. After that, the reason of we use firebase and Angular are both google technologies. In this way, we have revealed a long-term software.

Adaptability

The application we have developed is designed to adapt to changing requirements and changing requests. We designed our system for current conditions however when the requirements and conditions changing on time our system can change and update with our database.

Availability

Availability is the ratio of time a system or component is functional to the total time it is required or completion of the transaction in the expected time. The online attendance tracking system we developed is constantly online and we designed to allow users access anytime.

1.2. Interface Documentation Guidelines

Side Navigation

Our developed application has menu bar button. Users can pass the screens with the menu bar shortcuts.

Icons

We designed our system under stable and easy learn so we designed simple images for the describe functionality.

Buttons

While we were creating the buttons which we use in the application, we have used drawable and layout so with the html, css source codes design which are offered by Angular CLI. We have used particular colors during coloring. (All of the buttons are in the same color.)

1.3. Definitions, Acronyms, and Abbreviations

Acronyms

Some button names and ID's that developers can understand.

Abbreviation

There is no any abbreviation.

1.4. References

https://doodle.com/free-online-voting
https://www.easypolls.net

When The Ring Bells

2. **Package**

Class

- CourseAdd
- Profile
- ListAllCourses
- ListMycourses
- Login
- Register
- ForgotPassword
- Open Attendance
- Take Attendance
- ViewStudentList
- Editstudentinformation
- RequestSection
- UserService
- CourseService

CourseAdd

- **Veriable**

regiFrom
categories
teacher

- **Constructor**

CourseAdd() get userService, FormBuilder and CourseService

- **Methods**

OnSubmit() retrieves all necessary changes with form by html.
AddCourse() Send variables to service side.

- **Html**

On this side, the user enters the course, the teacher and the name of the course.

Profile

- **Veriable**

Name
Email

- **Constructor**

Profile() get userService

- **Methods**

seeDetail() Retrieves the current user's information from the service.

- **Html**

Displays their information to the user.

ListAllCourses

- **Veriable**

dataCourses
key

- **Constructor**

ListAllCourses() get userService and afAuth: AngularFireAuth
ngOnInit() get all courses

- **Methods**

getAllcoursess () get all courses from database

getReq() set the request for course.

- **Html**

The name of the lesson and the name of the teacher appear, next to it there is a button to join the lesson.

ListMycourses

- **Veriable**

dataCourses

- **Constructor**

ListMycourses()get userService and afAuth: AngularFireAuth
ngOnInit() get own courses

- **Methods**

getcoursess () get courses from database

- **Html**

The name of the lesson and the name of the teacher appear.

Login

- **Veriable**

data
info

- **Constructor**

Login() get userService

- **Methods**

getId() redirects the entries to the service side to login. It also informs the user if there is an incorrect user name or password entry.

- **Html**

It receives the username and password from the user and sends it to the component side.

Register

- **Veriable**

data
info

- **Constructor**

Register() get userService

- **Methods**

getInfo() receives the information required for registration, and sends them to the Service. warns the user if a wrong, missing, or missing user name is entered.

- **Html**

It receives the necessary information from the user for registration and sends it to the component side.

ForgotPassword

data
info

- **Constructor**

ForgotPassword() get userService

- **Methods**

getPassword() receives the information required for change password, and sends them to the Service. warns the user if a wrong things.

- **Html**

It receives the necessary information from the user for change password and sends it to the component side.

Open Attendance

- **Veriable**

regisFrom
dataCourses
DersinAdı
DersinOgretmeni
OgretmenId

- **Constructor**

OpenAttendance() get userService, FromBuilder and CourseService
ngOnInit() get all Teacher

- **Methods**

onSubmit() it gets veriable from HTML then pust it to service method

- **Html**

The user receives the name of the course and the instructor giving the course and sends it to the component side.

TakeAttandence

- **Veriable**

dataCourses
newDate
courseId
userTemp:firebaseUser
öğrenci
viewDetails

- **Constructor**

Courses() get userService , afAuth: AngularFireAuth , CoursesService and AngularFireDatabase
ngOnInit() get own courses

- **Methods**

getAttandance() This Method list students , get Today date

getReq() This method take attendance when teacher click var/yok

getAttDetails() When students click Show details, this method shows student's attandance details.

- **Html**

When the teacher enters this page, he sees the courses he has given and the students taking the course. The teacher can take attendance on this page.

When the student enters this page, he / she can see the lessons and attendance details.

ViewStudentList

- **Veriable**

  Datacourses
  İd
  Email
  userTemp:firebase.user

- **Constructor**

  usersProfil()get userService , afAuth: AngularFireAuth , CoursesService and AngularFireDatabase

  ngOnInit() get allStudentsProfil

- **Methods**

  Edit()send the student's 'id' to the service side and back it up for later use.

- **Html**

On this page all students can be viewed.

Editstudentinformation

- **Veriable**
  editForm
  email
  name
  id

- **Constructor**
  userEdit()get userService , FormBuilder, CoursesService and AngularFireDatabase

- **Methods**
  Edit() it sends the student's new information to the service side and changes it in the database.

- **Html**

on this page the student's information can be renewed.

RequestSection

- **Veriable**
  listArray
  listRequest
  userTemp:firebase.user

- **Constructor**
  requestSection()get userService , afAuth: AngularFireAuth , CoursesService and AngularFireDatabase

ngOnInit() arriving requests in the array and determines the role.

- **Methods**

applyRequest() this method approves the request and adds the student to the course in the database.

- **Html**

  On this page the student affairs and admin can see and approve requests from the student.

UserService

- **Methods**

isAdmin() checks whether the user is an admin.

isOgrenciIsleri() checks whether the user is student affairs

isOgrenci() checks whether the user is students

isOgretmen() checks whether the user is theacher
editProfil() updates the student's information.

Login() do the necessary tasks for login.

Logout() do the necessary tasks for logout.

getCurrentUser() retrieves the information of the logged-in user.

canActivate() prevents unauthorized users from entering pages.

singUp() does the necessary work for registration.

forgotPassword() does the necessary work to change the password.

CourseService

getAllCourse() brings the lessons under the teacher.

getAllCourseStudent() brings the lessons under the student.

getAllcourse() brings all lessons.

getOgretmen() brings all teacher.

getAdmin() brings all admin.

getItems() brings all request courses

sendRequest() the student sends the course request.

getAllStudentProfil() brings all teacher students.

ResponseRequest() confirms the request from the student.

ogrenciYap() makes the specified person a student.

ogretmenYap() makes the specified person a teacher.

ogreciIsleriYap() makes the specified person a student affairs.

AddCourse() Adds courses to the system.

getMyStudents() lists the students enrolled in the designated course.

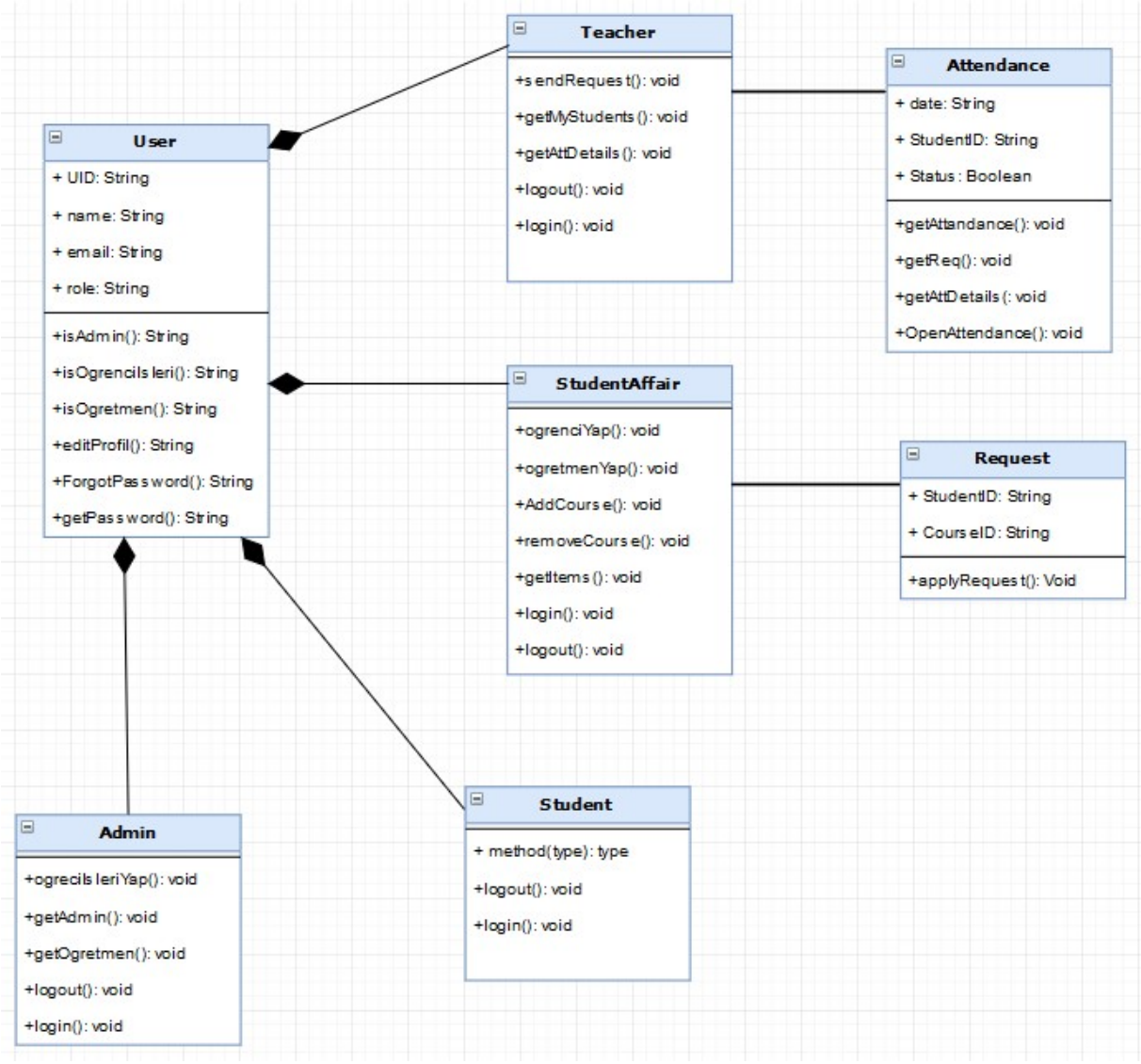
getAtt() polling takes.

removeCourse() deletes the specified course.

getAttDetails() shows the detail of the student polling.

3. Class Interfaces

Class Diagram



Database Diagram