



KOÇ UNIVERSITY

College of Engineering

**COMP 491 – Computer Engineering Design Project
Final Report**

KOLT TUTORING SYSTEM

Participant information:

**Deniz Yılmaz 69743
Doğa Demirtürk 68859
Sinan Cem Erdoğan 68912
Zeynep Gencer 68968**

**Project Advisor
Serkan Çil**

Spring 2023

Table of Contents

Table of Contents	2
1. Abstract	3
2. Introduction	3
3. System Design	5
3.1 Database	5
3.2 User Interface	7
3.3 User Roles	8
3.3.1 Staff Role/Pages	8
3.3.1.1 Users	8
a) Admin Users	8
b) Tutors	9
c) Head Tutors	11
3.3.1.2 Courses	13
3.3.1.3 Tutor Applications	14
3.3.1.4 Set Cubicles	15
3.3.1.5 Settings	18
a) Configuration	18
b) Schedule Texts	19
c) Cubicles	20
d) Periods	20
3.3.1.6 Generic Export Functionality	21
3.3.2 Tutor Role/Pages	22
3.3.2.1 My Page	22
3.3.3 Head Tutor Role/Pages	23
3.3.3.1 My Tutors	23
3.3.4 Student Role/Pages	24
3.3.4.1 Application	24
3.3.4.2 Cubicle Table	25
3.3.4.3 Schedule Table	25
3.4 Problems Encountered	26
4. Analysis and Results	27
5. Conclusion	28
6. References	29
7. Appendix	29

1. Abstract

The Koç University Office of Learning and Teaching (KOLT) Tutoring Center, established in November 2009, has a primary mission of providing free individual tutoring services to all Koç University students, ensuring continuous development in learning and teaching within the university. There are 22 undergraduates, 43 graduate, and 30 PhD programs with more than 700 courses offered in a term [1]. With numerous numbers of student, programs, and courses, organizing and managing the tutoring sessions pose significant challenges. The current system in place falls short in meeting these challenges effectively. It causes difficulties for admin users (KOLT staff), head tutors, tutors, and students in providing or attending tutoring sessions. This affects the quality and efficiency of the overall tutoring system at the university. The project offers a sleek and modern web application designed to facilitate the management and organization of all tutoring sessions, catering to four distinct user roles. Admin users have the ability to oversee the entire system and the other three user roles. Head tutors can view their assigned tutors, tutors can access their own information and view the period table to see their assigned tutoring periods, and students can access the schedule table, which provides information on the location, time, and course of each tutoring session. Overall, the application successfully addresses the challenges faced by the university's tutoring system.

2. Introduction

KOLT, the tutoring center at Koç University, offers free tutoring to all students. However, the current system used to manage and organise tutoring sessions, a web application [2], is problematic. The application was initially developed by a student who worked for KOLT under "Work and Study" program. Unfortunately, KOLT is currently facing challenges due to the lack of contact with the original implementer, and limited understanding of the system's functionality in the background. As a result, KOLT is unable to effectively address and resolve issues, leading to exhaustion among the KOLT office staff and a significant decrease in the overall tutoring system's efficiency in the university. Also, the user interface of the application old-fashioned and unresponsive.

The primary goal of the project is a new application with a modern and user-friendly interface and hand it over to the Koç University IT office. The IT office will take the responsibility of integration the application to Koç University's new platform KUHUB [3] and maintaining and supporting it. In this way, for any application related problem, KOLT office will be in touch with the IT office to solve it. The new application has a similar structure for the side of admin users. The outdated or unused pages and features removed. The three other user roles, head tutor, tutor, student, are defined and their pages added to the website. The user interface of the new application completely

redone.

The application shares similarities with existing scheduling systems like Doodle, Google Calendar, Outlook Calendar, etc. But requirements, features user role specific and unique to Koç University. Also, the user interface is unique to Koç University's new platform KuHub.

Google Calendar is a software where you can manage time scheduling. This software the users to determine and declare their availability to other users. Then other users can select among available time slots. Also, users can control their events and available times within Google software [4]. Estimated number of users of the software is over 500 million in 2021 [5].

Doodle is one of the most popularly utilized scheduling systems. Every month nearly 30 millions of users making pools on Doodle [6]. With a single link a user can create and share his/her available time with numerous of people. Any other user, even without a Doodle account can view created time slots and fill out the form with their name for the chosen time slot. After filling, a notification is sent to the user that created the time slot. The Doodle owner can control chosen time slots. Doodle make it easy to arrange meetings by eliminating one-to-one interaction.

Another popular scheduling software is Outlook Calendar which is provided by Microsoft. The software has a communication with the other services provided by Microsoft such as emails, contacts, and other features [7]. Using Outlook Calendar, a user adds events for a specific date and time in the calendar which is divided into time slots for days of the week. A user can select a slot as his/her choice and add an event for the slot or edit an existing event. Also, an event can be scheduled as recurring event. Time slots contain information about attendees and availability of them. For collective meetings, Outlook Calendar is a highly convenient and easy tool.

The application's time slot declaration and choosing parts is like that of traditional calendar systems, we have combined the parts we like from each various software systems and create a new time/course management tool. The management tool serves as an arranger between the four user roles of the system. While the core concept is like that of Doodle's, we have implemented additional features and developed a more detailed user interface. The application's newness comes from its characterised users such as the Admin User, Head Tutor, Tutor, and Students, and the unique roles of each of them. Unlike other applications, each role owner manages tasks and time slots through a separate page. The application is not just a time/course management tool, it is more than that. It has reviewing options and features like applying for becoming a KOLT Tutor, doing, exporting the current data, resetting some of the information regarding a semester and even assigning actual physical locations for meetings which are limited and can be modified as needed.

3. System Design

We created our system using C#, Javascript, HTML, and CSS. We used Visual Studio as IDE and PostgreSQL as our database. We will explain our database, user interface (UI), user roles and pages under their respective titles below.

3.1 Database

The first building block of our web application is our database. We used PostgreSQL as the database management system. We created a common database over Amazon Web Services (AWS) for all of us to access. In addition, we connected our application to the database by using Entity Framework in .Net MVC. Entity Framework is an object-relational mapping (ORM) framework used in .Net MVC applications to easily connect and manage the database. Thanks to the Entity Framework, users can work with databases using object-oriented concepts instead of writing SQL queries.

The name of our database is KuTutorCourseDb and it has 13 tables. These are AdminUsers, Configurations, Courses, Cubicles, EditScheduleTexts, HeadTutorConnections, HeadTutors, PeriodCubicles, Periods, TutorApplications, TutorCourses, TutorPeriods, Tutors.

Tutors table is created to store tutor information such as Name, Email, WeeklyHour and ID which is the primary key of this table. WeeklyHour attribute stores how many hours the tutor will teach per week. After the admin user accepts the student's application to become a tutor, the student's information is automatically stored in this table.

Courses table is created to store course information such as Code, Name and ID which is the primary key of this table. Only admin users can access this table and add, delete, or edit course information.

Cubicles table is created to store cubicle information. Cubicles are workspaces where the teacher teaches. This table has four attributes: CubicleName, CubicleNumber, CubiclePlace and ID which is the primary key of this table. CubiclePlace stores which building or office this cubicle is in. For example, cubicle place usually is KOLT. Only admin users can access this table and add, delete, or edit cubicle information.

AdminUsers table is created to store admin user's information such as Username, Email, FirstName, LastName and ID which is the primary key of this table. Only admin users can add, delete and edit data in this table.

Periods table stores the period information at which time the tutor will give the lesson. It has four attributes: Day, StartHour, EndHour and ID which is the primary key of this table. Only admin users can access this table and add, delete, or edit period information.

TutorApplications table stores the information the tutor entered when applying to become

a tutor. It has 8 attributes: Name, Email and GPA of the student, CourseIds, PeriodIds, WeeklyHour, Status and ID which is the primary key of this table. CourseIds, PeriodIds are list of integers that stores the IDs of the courses that the student wants to teach and the time slots that are available. In addition, Status attribute stores whether the admin accepts the application or not.

Configurations table is used by the admin to control whether the applications to become a tutor are seen by the student. It has 6 attributes: ID which is the primary key of this table, PublishSchedule which is a Boolean, Date, Time, EditDateTime and Username who created this configuration. Date and Time attributes store the application deadline. If PublishSchedule is true, it means that applications are open to students. As long as PublishSchedule is true and the deadline has not passed, students can apply to become a tutor. EditDateTime stores the date the user last edited the configuration.

TutorCourses table is an intermediate table that connects tutor and course entities. There is many-to-many relationship between them, and this relationship is that the tutor teaches course. It has 2 primary keys: TutorId and CourseId. When a tutor matches a course, the id of the tutor and the id of the course are automatically saved in the table.

TutorPeriods table is an intermediate table that connects tutor and period entities. There is many-to-many relationship between them, and this relationship is that the tutor teaches during that period. It has two primary keys, TutorId and PeriodId, and one simple attribute, PeriodTutorId which is the ID of the table itself. When a tutor is paired with a period, the id of the tutor and the id of the period are automatically saved in the table.

PeriodCubicles table is an intermediate table that connects TutorPeriods and Cubicles tables. There is one-to-many relationship between them, and this relationship is that the tutor teaches in this cubicle. It has one primary key, PeriodCubicleId which is the ID of the table itself, and two simple attributes, PeriodTutorId and CubicleId. When a tutor is paired with a cubicle in a specific period, the id of the TutorPeriods table and the id of the cubicle are automatically saved in the table. The assignment of a tutor to a cubicle in the desired period is performed automatically with for loops in the code. If there is a conflict after matching, that is, if 2 tutors are placed in the same cubicle in the same time zone, the admin can manually change the period of the tutor and assign it to the new cubicle via the application.

EditScheduleTexts table stores the text content of the schedule table. Admin users can set the content within the fixed fields of the schedule table. This table has 7 attributes: MainTitle, SubTitle, BelowTable, PageTitle, GoToTop, NotPublishedText and ID which is the primary key of this table. MainTitle stores the main title of the schedule table, SubTitle stores the information of the sub-title, BelowTable stores the descriptive content provided beneath the tables. PageTitle stores the title of every page. GoToTop stores the text for the link that takes the user to the top of the

schedule table. NotPublishedText stores the text that appears when schedule is not published.

HeadTutors table stores head tutor information such as HeadTutorTutorsId which is the tutor ID of this head tutor, TutorIds, CourseId which is the ID of the course the tutor will teach. TutorIds are the list of integers that store ID of the tutors assigned under supervision of this head tutor.

HeadTutorConnections table is an intermediate table that connects Tutor and HeadTutors entites. There is one-to-many relationship between them. A tutor can only have one head tutor for a course, but a head tutor can be responsible for more than one tutor. This table has 2 simple attributes, HeadTutorId which is the ID of this head tutor and TutorIds which are IDs of the tutors assigned under supervision of this head tutor. The primary key of this table is ID attribute of this entity.

3.2 User Interface

One of our main building blocks is the UI of our web application. Since we implemented our project with ASP.NET, we used Razor Pages with the Model-View-Controller approach. We designed our UI according to the UI/UX Kit provided by IT to match with the new KUHub platform. We designed our application in a responsive manner to make it compatible with different viewport sizes.

The application starts with a “Sign In” page which authenticates and signs in the user. The authentication and verification part is provided by IT to have the KUHub projects with the same infrastructure. We use the client information in our program.

The overall view of the application is implemented within the “Layout” page consisting of four main components: a sidebar, a top navigation bar, a footer, and the main content. Sidebar includes the main navigation of the application consisting of Student, Tutor, Head Tutor, and Staff pages, and the logo of the KUHub. Sidebar navigation content depends on the role of the signed user. The top navigation bar has the “Logout”, “Home”, and “Profile” links. Footer content is designed according to the provided kit. The main content part of the Layout displays the current page content with Razor syntax using the RenderBody function.

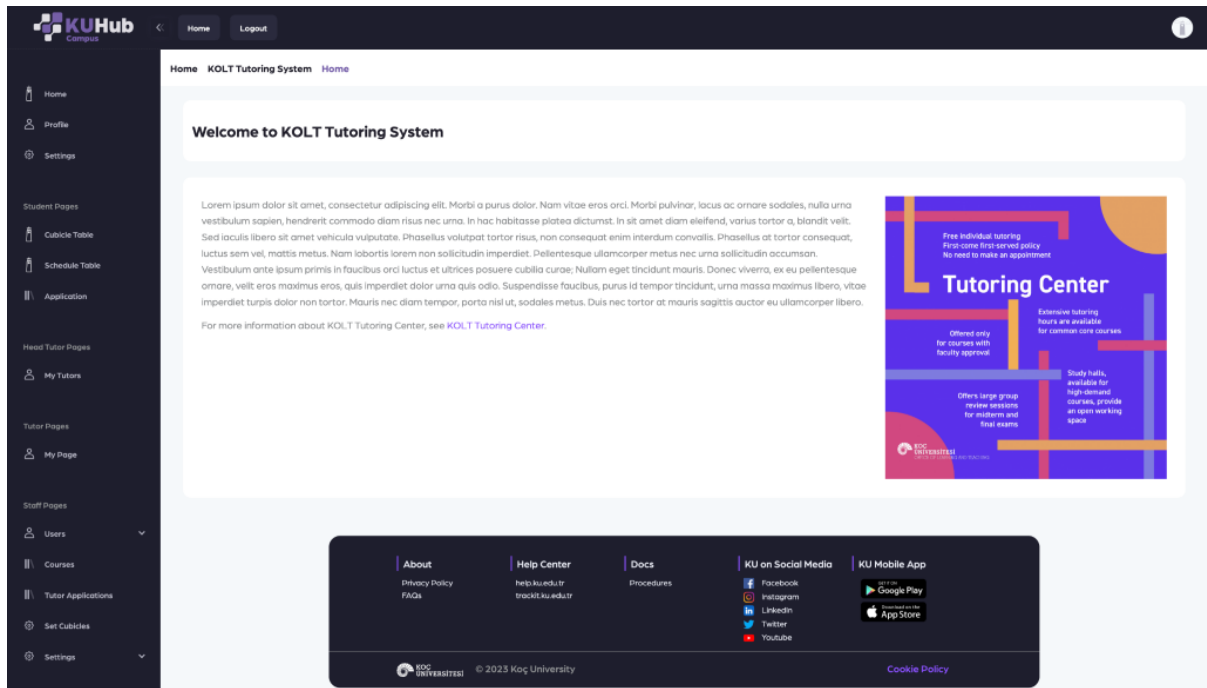


Figure 1 Home Page of the application

3.3 User Roles

3.3.1 Staff Role/Pages

3.3.1.1 Users

a) Admin Users

Admin users page is for role activation of Staff Users. When added to this table the user with the specified e-mail can acquire the staff role and see the staff pages. With this page, authentication will be easier and automated.

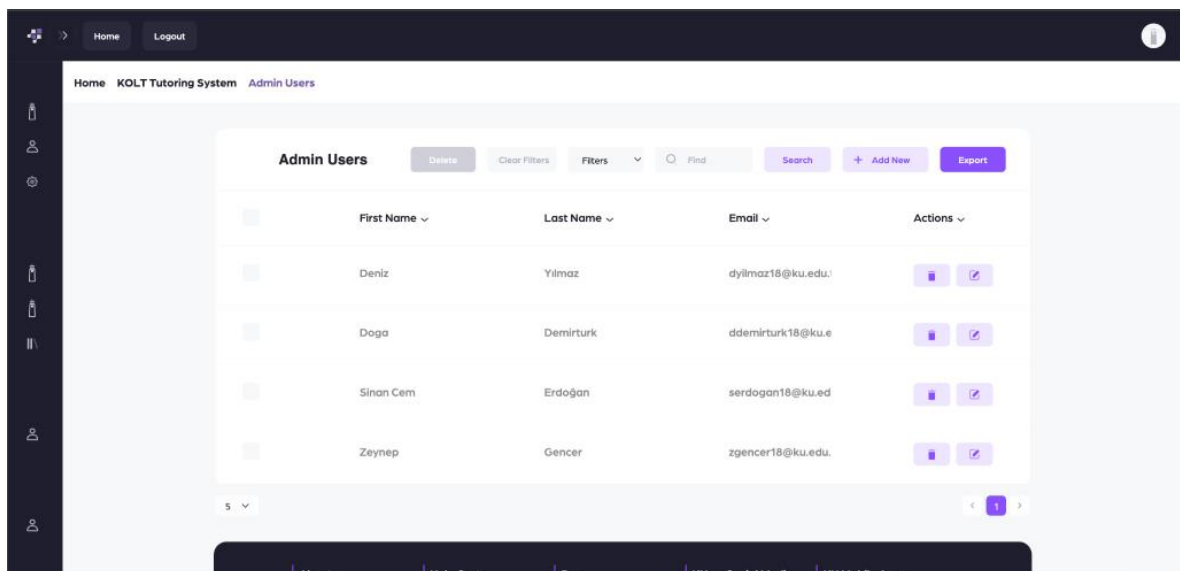


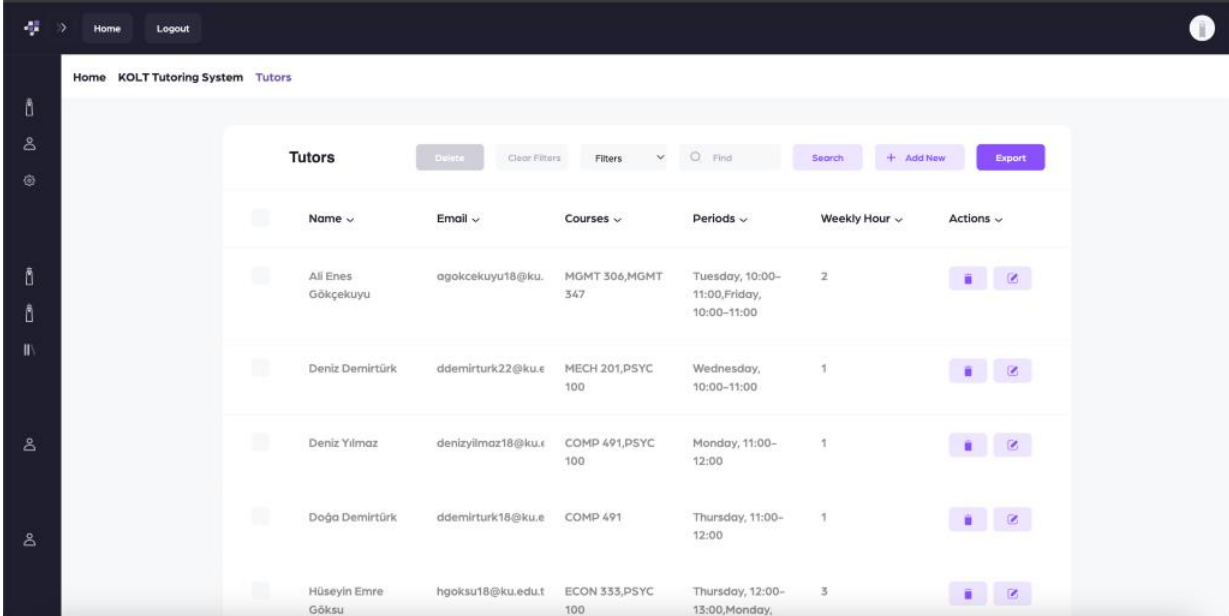
Figure 2 Admin Users Page

Only users with a staff role can add, edit and delete admin users. With add button; by specifying the name, surname, user name, and unique ku email, a new admin user is created (see Appendix, Figure 30). Although we can see the addition to this table right now, we are not able to assign staff roles to the new user due to Koç University privacy rules and Key clock actions that we are not authorized to. The addition to the table is done but the functionality is to be implemented by the Koç IT department.

The admin user page also has an edit functionality which enables staff to edit each information of the admin user on a new view page, and delete functionality to unauthorize the user from its staff roles which shows another view page with the user data before deletion and an option to cancel the action (see Appendix, Figure 31). This authorisation functionality is also not implemented due to above reasons.

b) Tutors

Tutors page is for role activation of Tutor Users. When added to this table the user with the specified e-mail can acquire the tutor role and see the tutor pages. With this page, authentication will be easier and automated. Although tutors can be applied and approved with the application, we implemented this page for the manual addition of tutors.













Name	Email	Courses	Periods	Weekly Hour	Actions
Ali Enes Gökçekuyu	agokcekuyu18@ku.	MGMT 306,MGMT 347	Tuesday, 10:00-11:00, Friday, 10:00-11:00	2	 
Deniz Demirtürk	ddemirturk22@ku.e	MECH 201,PSYC 100	Wednesday, 10:00-11:00	1	 
Deniz Yılmaz	denizyilmaz18@ku.e	COMP 491,PSYC 100	Monday, 11:00-12:00	1	 
Doğa Demirtürk	ddemirturk18@ku.e	COMP 491	Thursday, 11:00-12:00	1	 
Hüseyin Emre Gökse	hgoksu18@ku.edu.t	ECON 333,PSYC 100	Thursday, 12:00-13:00, Monday,	3	 

Figure 3 Tutors Page of the application

Only users with a staff role can add, edit and delete tutor users. With add button; by specifying the name, courses, periods, and unique KU email, a new tutor user is created. Although we can see the addition to this table right now, we are not able to assign tutor roles to the new user due to the above-mentioned reasons.

While adding the tutor, specifying the courses and periods are critique actions since they affect many other pages and functionalities and connections to the database. Each period and course is taken from the period and course entities. Thus, by adding the course to be given by the tutor is assigned to the tutor; a new connection is created at the database side on the TutorCouses table with their respecting primary keys. Similarly, when tutors lecturing periods are specified a new connection is established between that tutor and period by creating a new entry on database table TutorPeriods with their primary keys.

Weekly hours shown on addition are automated and show the number of periods that tutor lectures in a weekly manner.

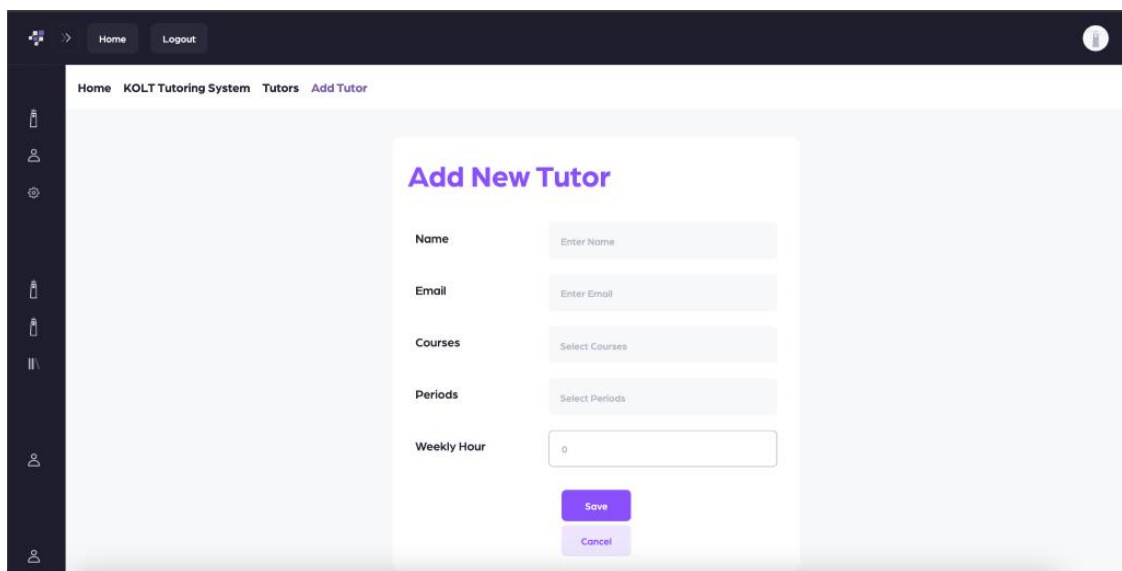
The image shows a web application interface for adding a new tutor. At the top, there is a dark navigation bar with 'Home' and 'Logout' links. Below this, a breadcrumb trail shows 'Home > KOLT Tutoring System > Tutors > Add Tutor'. The main content area features a light blue sidebar with icons for various system functions. The central focus is a white form titled 'Add New Tutor' in purple. The form contains five input fields: 'Name' (placeholder 'Enter Name'), 'Email' (placeholder 'Enter Email'), 'Courses' (placeholder 'Select Courses'), 'Periods' (placeholder 'Select Periods'), and 'Weekly Hour' (placeholder '0'). At the bottom of the form are two buttons: a purple 'Save' button and a light blue 'Cancel' button.

Figure 4 The 'Add New Tutor' functionality of the Tutors Page

The tutor user page also has an edit functionality which enables staff to edit each information of the tutor user on a new view page, and delete functionality to unauthorize the user from its tutor roles which shows another view page with the tutor data before deletion and an option to cancel the action (see Appendix, Figure 32). This authorisation functionality is also not implemented due to above reasons.

Edit Tutor

Name: Ali Enes Gökçekuyu

Email: agokcekuyu18@ku.edu.tr

Courses: MGMT 306, MGMT 347

Periods: Tuesday, 10:00-11:00, Friday, 10:00-11:00

Weekly Hour: 2

Save Cancel

Figure 5 The 'Edit Tutor' functionality of the Tutors Page

c) Head Tutors

Head Tutors page is for role activation of Head Tutor Users. When added to this table the user with the specified e-mail can acquire the head tutor role and see the head tutor pages. With this page authentication will be easier and automated.

Head Tutors Delete Clear Filters Filters Find Search + Add New Export

	Course	Head Tutor	Tutors	Actions
	COMP 491	Doğa Demirtürk	Deniz Yılmaz	
	PSYC 100	Zeynep Gencer	Naz King, Deniz Demirtürk, Hüseyin Emre Göksu	

5 < 1 >

Figure 6 Head Tutors Page of the application

Only users with a staff role can add, edit and delete head tutor users. With the add button; by specifying the course given by the tutor, tutor to be assigned as head tutor, and tutors assigned to that head tutor, a new head tutor user is created. Although we can see the addition to this table right now, we are not able to assign head tutor roles to the new user due to above-mentioned reasons.

Each head tutor is selected among the tutors. There is no manual addition of a head tutor if there is not an entity of that tutor. When the course is selected, only the tutors who give that course

are listed. We achieved this functionality by filtering the tutors to be listed at the controller side. The GetFilteredTutors function is triggered from the view via AJAX. After selecting the head tutor from this filtered tutors list, tutors to be assigned to that head tutor are shown on the page with the head tutor removed from the list.

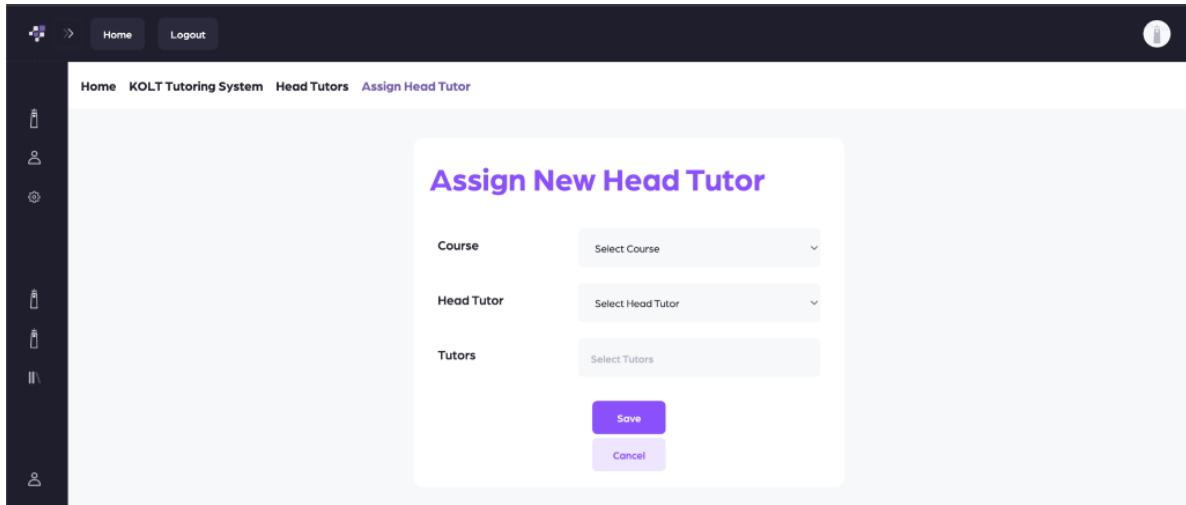
The screenshot shows a web application interface for the 'KOLT Tutoring System'. The top navigation bar includes 'Home' and 'Logout' links. A sidebar on the left contains icons for various system functions. The main content area displays the 'Assign New Head Tutor' form. This form has three dropdown menus: 'Course' (labeled 'Select Course'), 'Head Tutor' (labeled 'Select Head Tutor'), and 'Tutors' (labeled 'Select Tutors'). Below these dropdowns are two buttons: a blue 'Save' button and a light blue 'Cancel' button. The breadcrumb trail at the top of the content area reads 'Home > KOLT Tutoring System > Head Tutors > Assign Head Tutor'.

Figure 7 The 'Assign New Head Tutor' functionality of the Head Tutors Page

The head tutor user page also has an edit functionality that enables staff to edit tutors assigned to that head tutor on a new view page, and delete functionality to unauthorize the user from its head tutor roles which shows another view page with the head tutor data before deletion and an option to cancel the action (see Appendix, Figure 33 and 34). This authorisation functionality is also not implemented due to the above reasons.

3.3.1.2 Courses

The Courses page lists all the courses available in the application. The listed courses can be filtered by course code or name, and users can search for specific courses using the filter. There are two ways to add courses to the table. The first method is through the import functionality. At the top right corner of the table, there is a button called “Import.” When users click this button, the application sends a request to <http://open-api.ku.edu.tr/> to retrieve the courses offered in the specified term selected by the user.

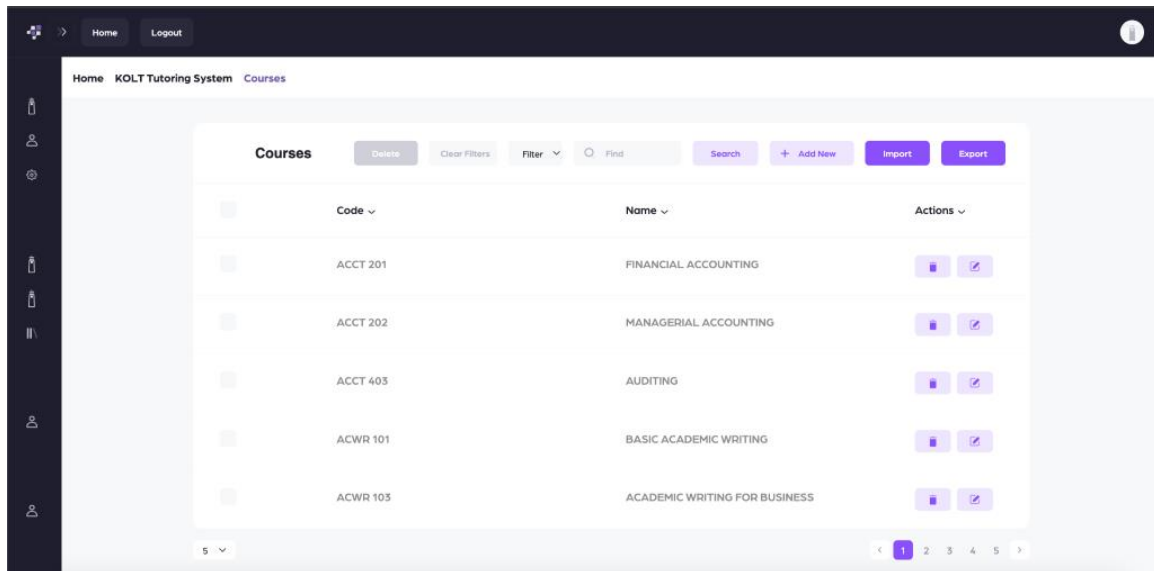


Figure 8 Courses Page of the application

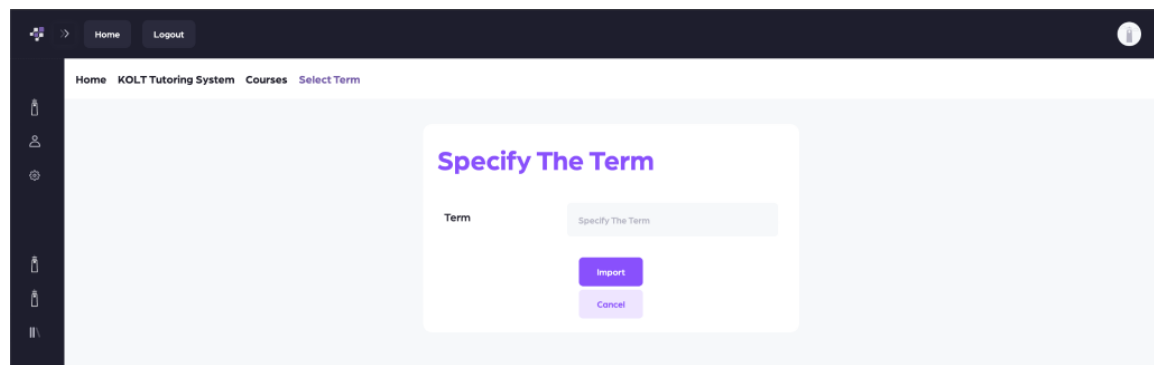


Figure 9 The 'Import' functionality of the Courses Page

The other way is to use the "Add New" functionality. Located at the top right corner of the table, there is a button named "Add New." By clicking this button, users can manually add a new course to the table by providing the course code and name (see Appendix, Figure 35). Users can also edit a course from the table by clicking the edit button in the actions column. This allows users to modify the course code and name (see Appendix, Figure 37). Lastly, a course can be removed from the table by clicking the trash icon in the actions column or by selecting multiple courses using the checkboxes on the left side of the table and clicking the "Delete" button (see Appendix, Figure 36).

3.3.1.3 Tutor Applications

Tutor Applications page contains a table filled with the applications submitted by students to become tutors from the "Application" page available for all students of Koc University. Staff can filter and search applications by their attributes. Instances can be edited and deleted from the actions buttons at the end of the rows (see Appendix, Figure 45 and 46). By clicking the "Add New" button, a new form page where the staff can add a new tutor application is opened.

Name	Email	Courses	GPA	Periods	Weekly Hour	Status	Actions
Begüm Şen	begumsen19@ku.edu.tr	COMP 301	3,5	Friday, 10:00–11:00	1	Pending	[+], [trash], [edit]
Defne Ersavaş	dersavas18@ku.edu.tr		3,5	Wednesday, 10:00–11:00	1	Denied	[trash], [edit]
Deniz Yılmaz	denizyilmaz18@ku.edu.tr	PSYC 100	3,67	Tuesday, 10:00–11:00	1	Approved	[trash], [edit]
Sinan Cem Erdoğan	serdogan18@ku.edu.tr		3,09	Monday, 10:00–11:00	1	Denied	[trash], [edit]

Figure 10 Tutor Applications Page of the application

Moreover, staff can approve or deny the pending applications. When approved, a new tutor instance with the information from the approved tutor application is created and added to the database. From now on the applicant will have the tutor role and will be able to see its tutor page. Multiple tutor applications can be accepted, rejected, or deleted by selecting the desired rows from checkboxes using the buttons at the top of the table.

Approve or Deny Tutor Application

Name: Begüm Şen

Email: begumsen19@ku.edu.tr

Courses: COMP 301

Periods: Friday, 10:00–11:00

Weekly Hour: 1

Approve

Deny

Cancel

Figure 11 The 'Approve or Deny Tutor Application' functionality for admin users

3.3.1.4 Set Cubicles

Set cubicles have more than one functionality. When pressed, the table showing the weekly view of the Kolt Tutoring Center plan appears. This table shows only the periods that are filled by tutors at the time. Filled means that period is selected by a tutor. Periods are listed on a daily basis and each hour period is separated. At each time period there exists numbers linked to Tutors page. These numbers give the information regarding the number of tutors which give lectures at that

period. When pressed the numbers link the user to tutors page where the tutors are filtered by the specified period. Meaning only the tutors who selected that period are listed on Tutor page.

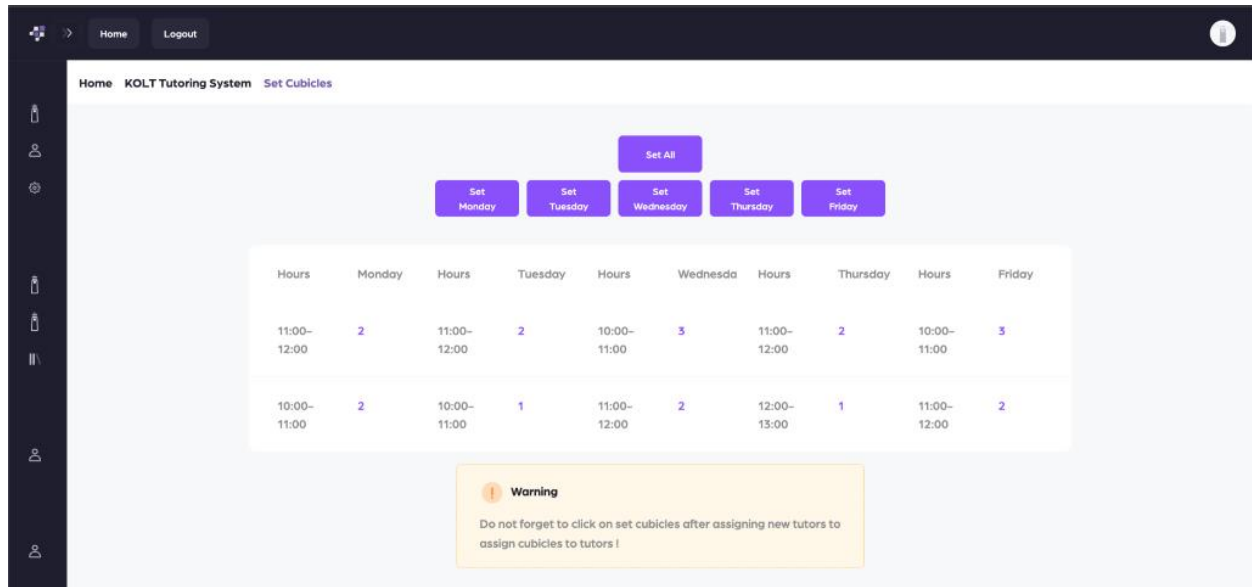


Figure 12 Set Cubicles Page of the application

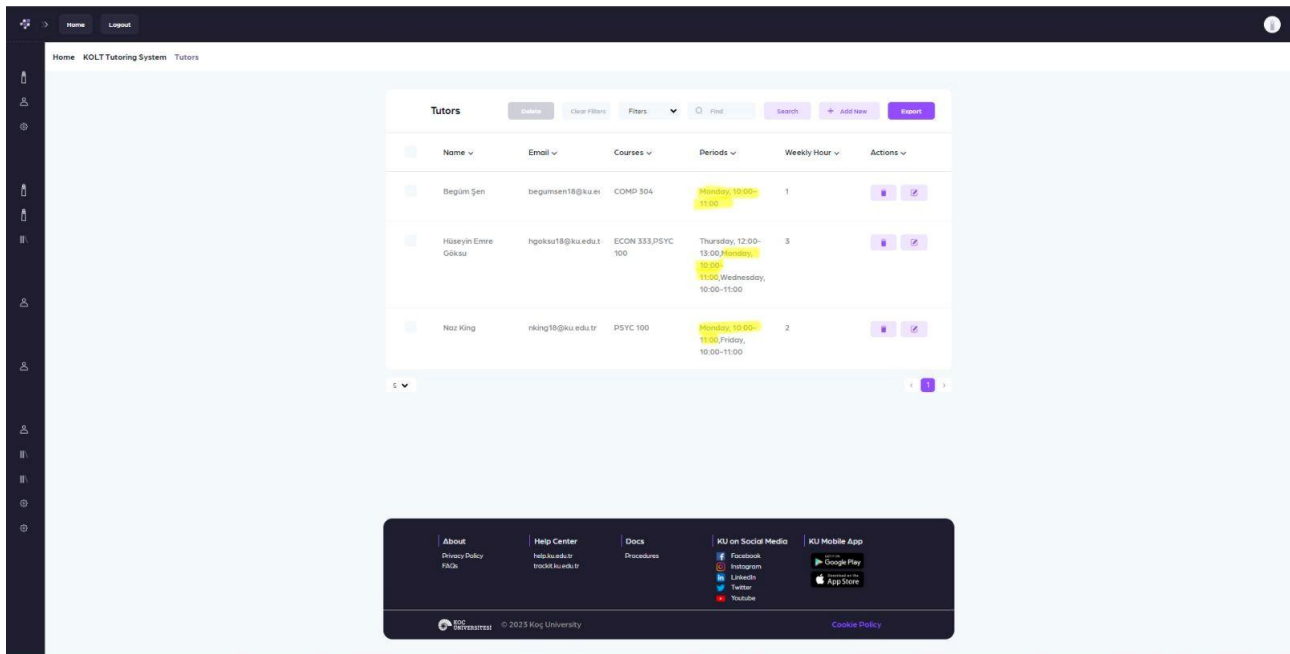


Figure 13 When the admin approves an application, that student automatically becomes a tutor.

There exists a warning at this page. Warning indicates “Do not forget to click on set cubicles after assigning new tutors to assign cubicles to tutors”. This warning is meant to lead the user to above buttons. Each button works for the days they are named after. Set all button is an overall setting button. Setting means for setting the cubicles for tutor. It works like a google calender.

When tutors select new time periods, admin users add new tutors, edit them or delete them this set functionalities needs to be run in order to that change to be effective on cubicle plans.

Cubicle plans are created by tutor period and course informations. Each tutor needs to teach at a specified location. These locations are not automatically assigned to tutors due to their dynamic nature. When tutor/admin choose/changes their periods and the set buttons are pressed, set cubicle algorithms take place.

Each tutor period has a connection to cubicle table on database. These connection information is held on PeriodCubicle table with their primary keys. But entries to this table are not manual, they are automated from the set cubicle buttons.

When pressed, algorithm checks on Tutor periods and chooses the ones which are not assigned paired up with cubicles on PeriodCubicle table; take this period tutor connection and extract tutor and period informations separately; then algorithm checks cubicles at that specified time period; cubicles are traversed regarding their id; if the cubicle is free, meaning there is no other tutor period assigned to it in PeriodCubicle table we assign this period and tutor data to this cubicle and add the connection to Period cubicle table; else if the cubicle is not available next cubicle is checked and so on. If all cubicles at that period are occupied, the tutor is assigned to the last cubicle in the list. In this case conflict occurs and table ui changes.

On the set cubicles table, we colour coded the conflicts and desired plans. When the cubicle is free we used a yellow/orange colour for the cell to emphasise emptiness. When there is exactly one tutor at a specific cubicle at a specific period we coloured the cell with blue/green. When there exists more than one tutor at a cubicle, meaning a conflict occurred, we showed the cell with a red colour.

Session Hours → Cubicle No ↓	10:00 – 11:00	10:59 – 12:59	11:00 – 12:00
Cubicle 1	Zeynep Lara Karadoğlu NTL 201, INTL 501		Sadık Muhammet Gencer BUSA 499, LAW 106
Cubicle 2	Ali Eneş Gökekeşçi MOMT 306, Naz King PSYC 100		Zeynep Gencer ARHA 122, Berat Karayilan COMP 304

Figure 14 After the cubicle assign operation, conflicts may occur.

The set cubicles table has a drag drop functionality to manually change tutor periods and cubicles. Especially when a conflict occurs, to change the conflicting tutor's period and/or cubicle just pressing the mouse on the tutor and dragging the tutor to desired cell and dropping tutor to that cell is enough to change tutor's period and cubicle information. This action effects tutor entity, tutor

period and period cubicle connection tables.

Session Hours→ Cubicle No↓	10:00 – 11:00	10:59 – 12:59	11:00 – 12:00
Cubicle 1	Zeynep Lara Kadioğlu INTL 201, INTL 301	Naz King PSYC 100	Sadık Muhammet Gencer BUSA 499, LAW 106
Cubicle 2	Ali Enes Gökçekuyu MGMT 306, Naz King PSYC 100		Zeynep Gencer ARHA 122, Berat Karayılan COMP 304

Figure 15 The admin users can resolve conflicts by drag and drop.

Session Hours→ Cubicle No↓	10:00 – 11:00	10:59 – 12:59	11:00 – 12:00
Cubicle 1	Zeynep Lara Kadioğlu INTL 201, INTL 301	Naz King PSYC 100	Sadık Muhammet Gencer BUSA 499, LAW 106
Cubicle 2	Ali Enes Gökçekuyu MGMT 306, MGMT 347		Zeynep Gencer ARHA 122, Berat Karayılan COMP 304

Figure 16 The admin users can resolve conflicts by drag and drop.

The setting of cubicles for periods, changing tutor periods and cubicles also effects several view pages. Each change is applied to their respecting view. Other than staff pages, tutor, head-tutor and student pages are effected due to the mass data change.

3.3.1.5 Settings

a) Configuration

Configuration page is used to set or edit the configuration elements of the overall system. These elements include “PublishSchedule,” which indicates whether the schedule table is published or not, “Date,” which represents the deadline date for tutor applications in the format of day, month, and year, and “Time,” which represents the deadline time for tutor applications in the format of hour and minute. There is only one Configuration object for the entire application. Once it is created, users are allowed to edit it (see Appendix, Figure 38). The edit details and the users who made the edits can be viewed in the history (see Appendix, Figure 39).

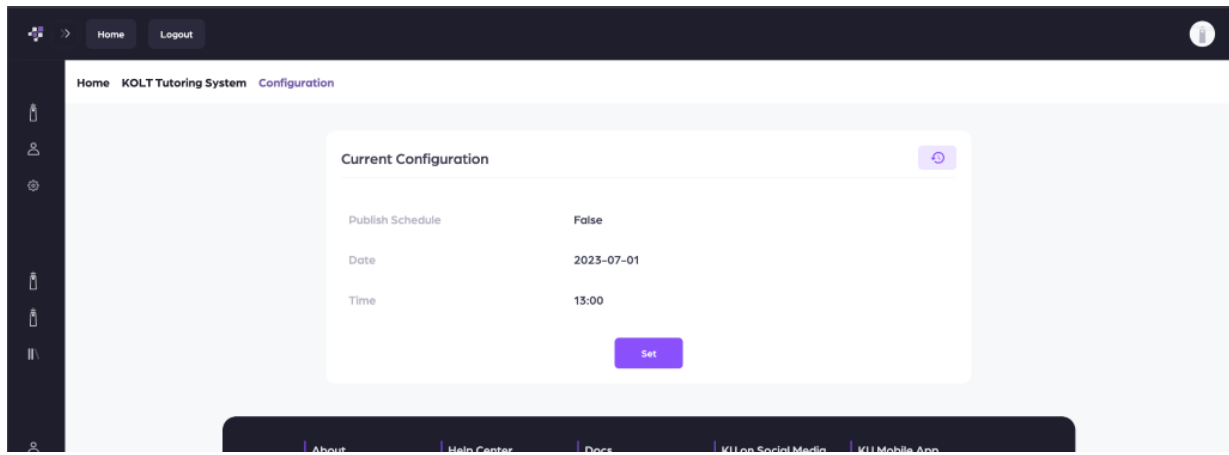


Figure 17 Configuration Page of the application

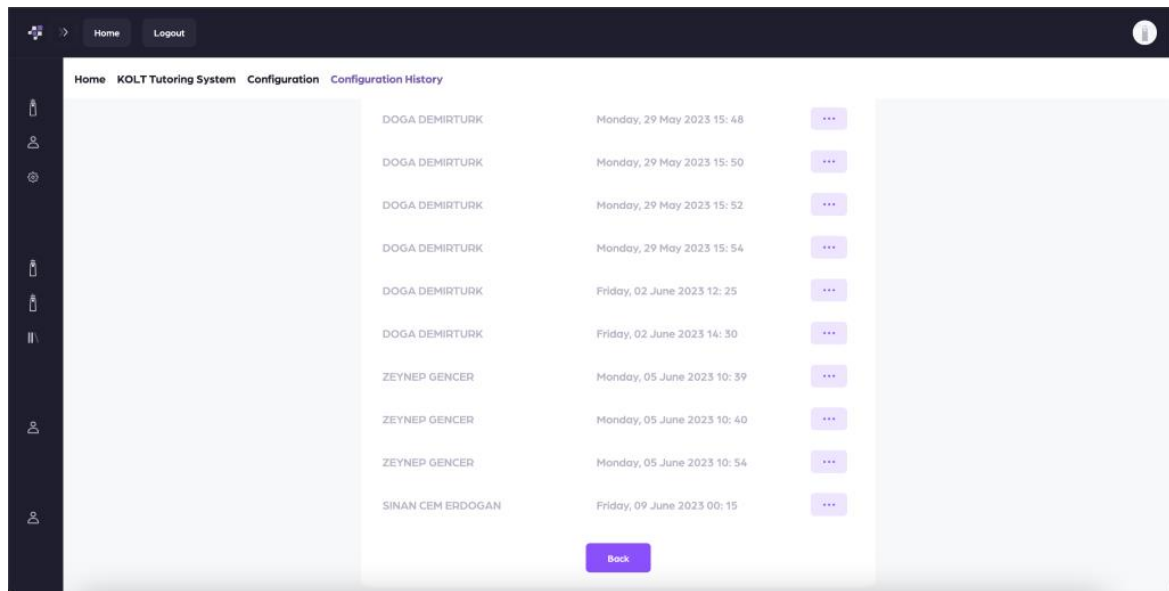


Figure 18 Configuration History Page of the application

b) Schedule Texts

The purpose of Schedule Texts page is to set the text content within the fixed fields of the schedule table. These fields are named “MainTitle,” “SubTitle,” “BelowTable,” “PageTitle,” and “GoToTop,” and “NotPublishedText.” “MainTitle” sets the main title of the schedule table, “SubTitle” sets the sub-title, “BelowTable” sets the informative text below the tables, “PageTitle” sets the title for every page, “GoToTop” sets the text for the link that takes the user to the top of the schedule table, “NotPublishedText” set the text that appears when schedule is not published . There is only one Schedule Texts object for the entire application, and once created, users are allowed to edit it.

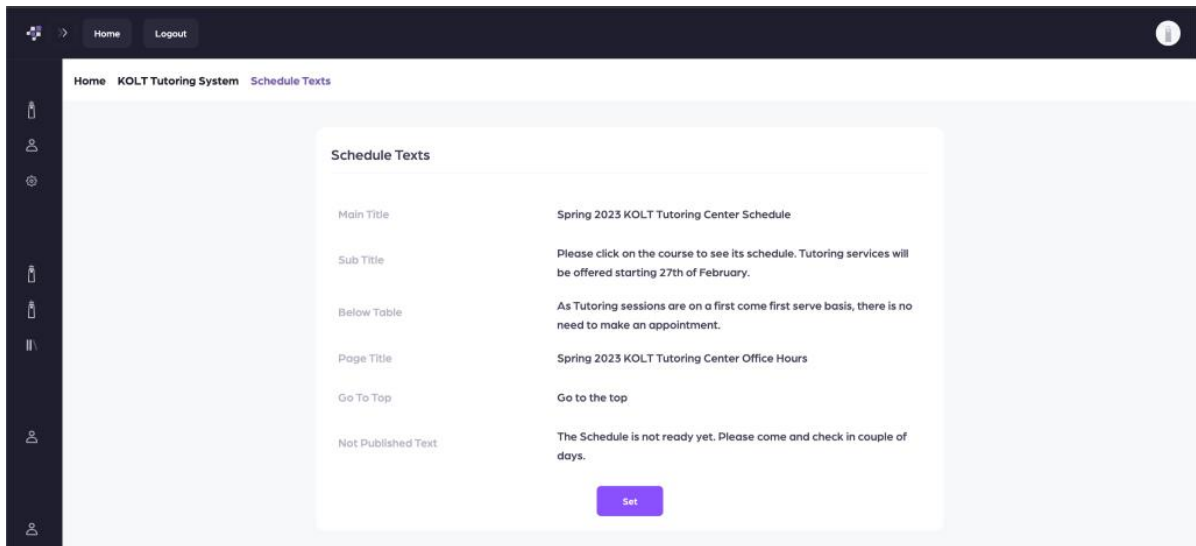


Figure 19 Schedule Page of the application

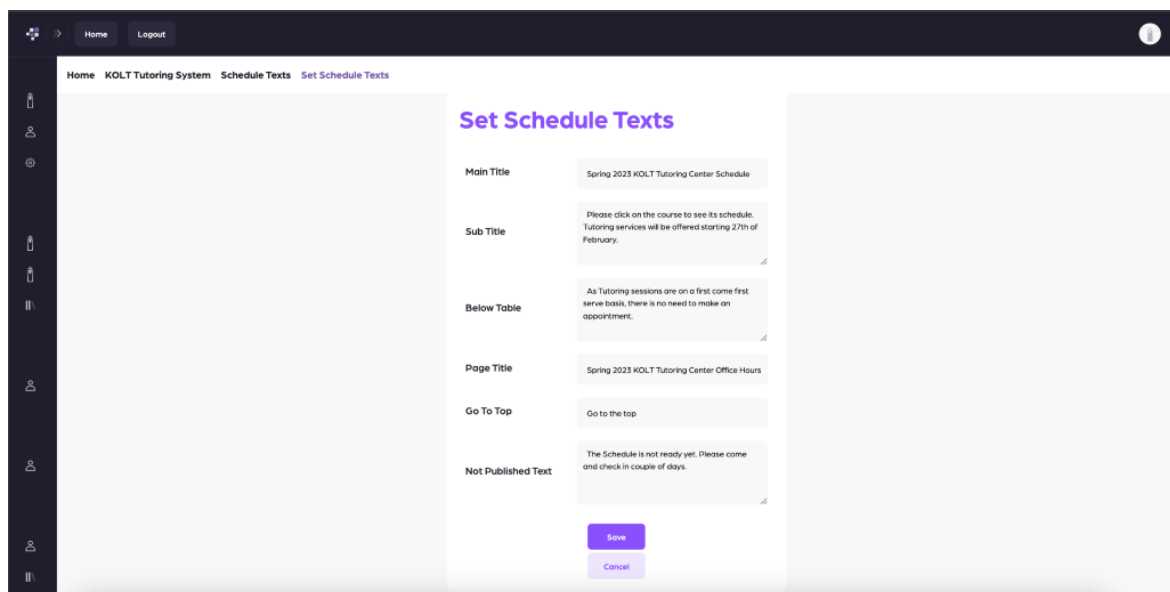


Figure 20 The admin users can set schedule text.

c) Cubicles

Cubicles page is for adding, editing and deleting place information of teaching action of tutors. These entities are connected with tutor and periods later on. Although they don't have much functionality their importance is crucial.

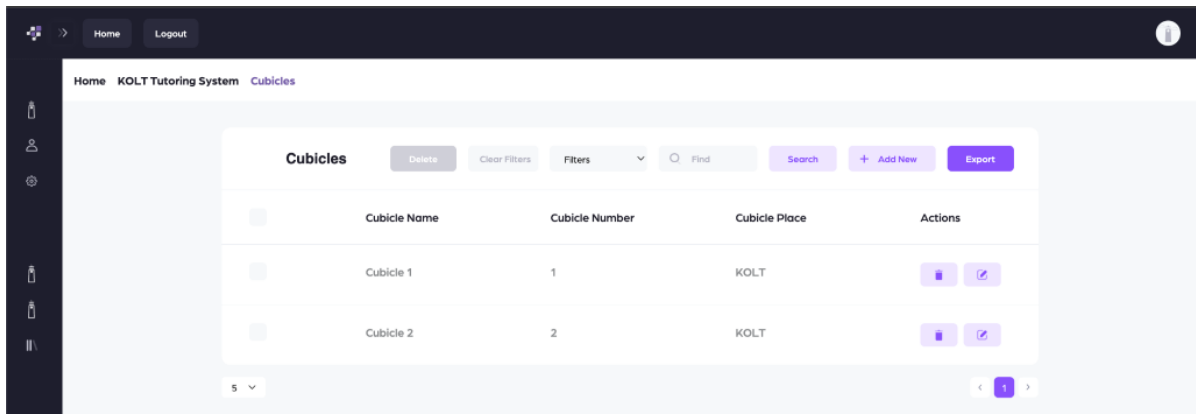


Figure 21 Cubicles Page of the application

A new cubicle can be added by specifying its name, number, and place (see Appendix, Figure 40). With edit and delete functionalities this information can be managed (see Appendix, Figure 41 and 42).

d) Periods

Period page is for adding, editing and deleting time information of teaching action of tutors. These entities are connected with tutors and cubicles later on.

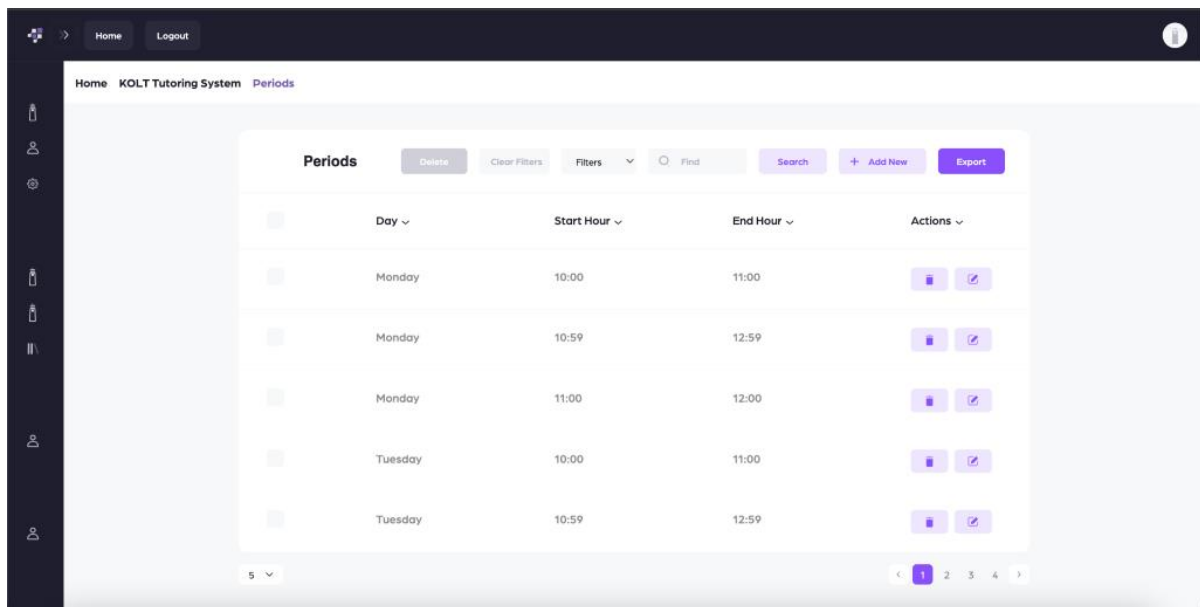


Figure 22 Periods Page of the application

Periods can be added either for individual days or for all weekdays. After choosing a day from a combo box, start and end hours and minutes can be selected from a time picker. While selecting start hour must be smaller than the end hour for avoiding warnings and being able to create the period object.

Figure 23 The admin users can add new period.

Edits and deletes for the periods are done individually (see Appendix, Figure 43 and 44). For each edit times and days can be changed.

3.3.1.6 Generic Export Functionality

There is a generic export functionality available for the following tables: "AdminUsers," "Tutors," "HeadTutors," "Courses," and "TutorApplications." For each of these tables, there is a button labeled "Export" located at the top right corner. When this button is clicked, the data is retrieved from the database for the respective table, converted into a CSV file, and downloaded to the user's computer.

3.3.2 Tutor Role/Pages

3.3.2.1 My Page

Tutors are users that hold the tutor role. This role can be acquired either by getting an approval from student's tutor application or addition to tutor page from the staff pages side directly. The tutor page named "My Page" shows the tutor user the information that belongs to them. The information shown includes name, email, courses, periods and weekly hour.

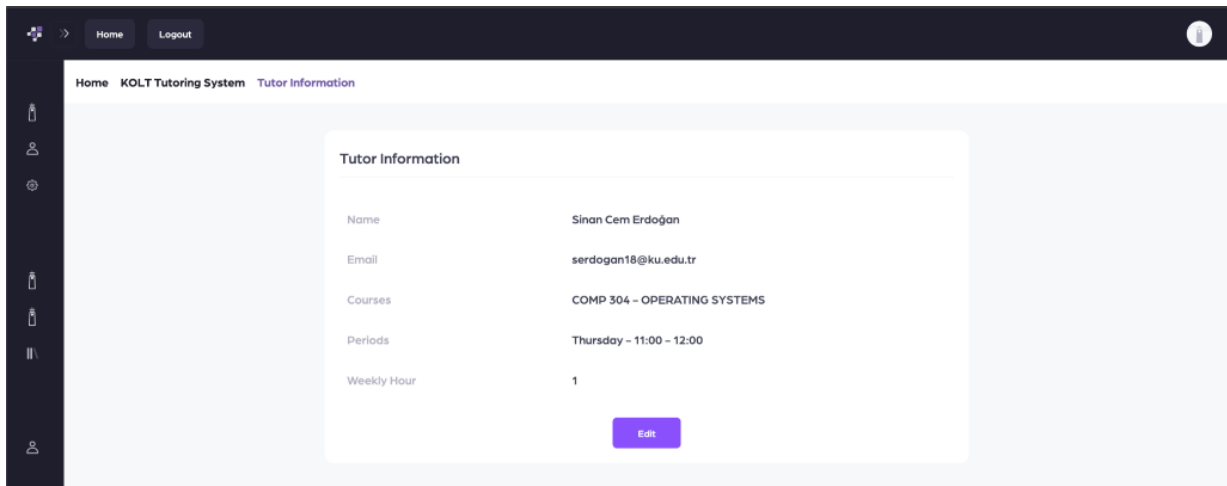


Figure 24 Tutors can see their tutoring information on their ‘My Page’ page.

Tutors teach at KOLT work on period hour basis. After choosing their courses at the beginning of the semester, they just change the teaching periods to match the monthly hour limit. Tutors can change the periods they teach at the beginning of each month or week. That's why on the edit page of My Page periods of the tutor can be changed by the tutor. This period information can be seen and set by the manager later on.

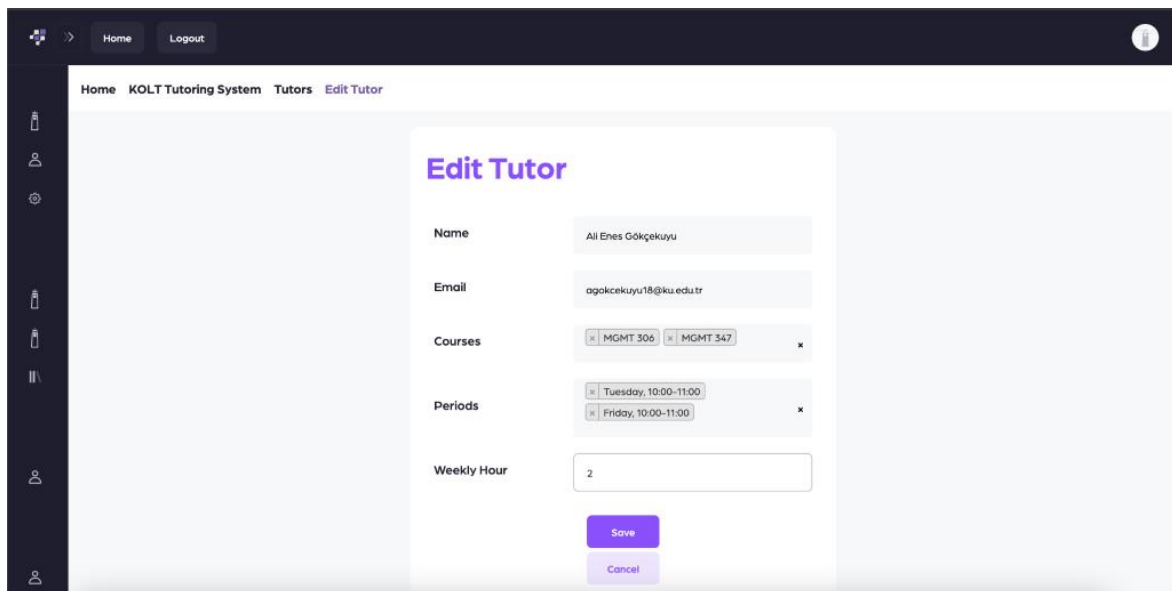


Figure 25 Tutors can edit their information.

3.3.3 Head Tutor Role/Pages

3.3.3.1 My Tutors

Head tutors are users that hold the head tutor role. Only an existing tutor can be assigned to a head tutor role. Head tutor roles can be acquired after a staff user adds a head tutor. The head tutor page “My Tutors” shows the assigned tutors of the head tutor. Assigned tutors are listed as a table and their name, email, course, period and weekly hour informations are shown. Head tutors does

not have any authorization over these information they can only view the information.

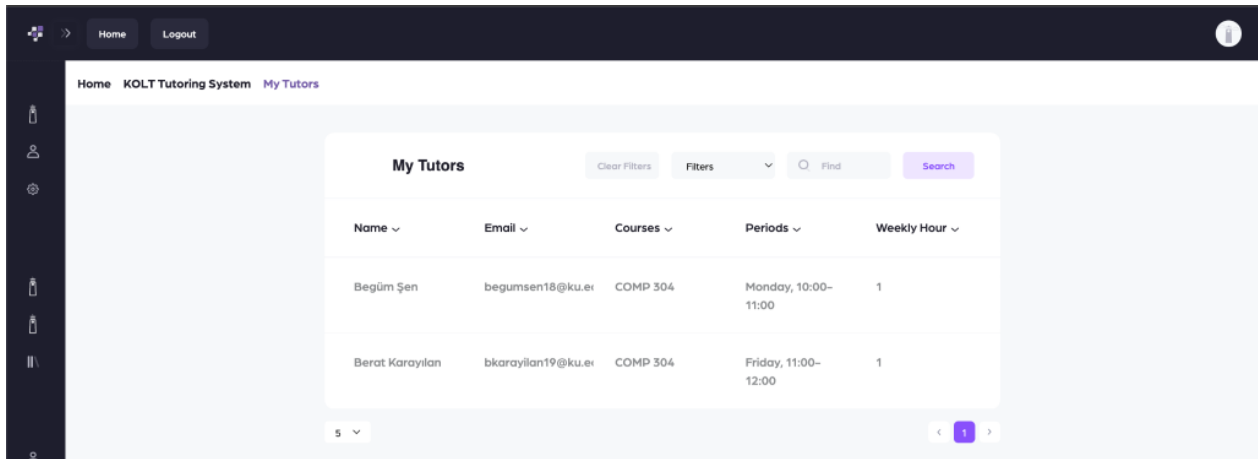


Figure 26 Head Tutors can see tutors that are assigned under their supervision on their ‘My Tutors’ page.

3.3.4 Student Role/Pages

3.3.4.1 Application

Every student can apply to become a tutor if the time is before the deadline period of application. When the time is right an application button is shown on this page.

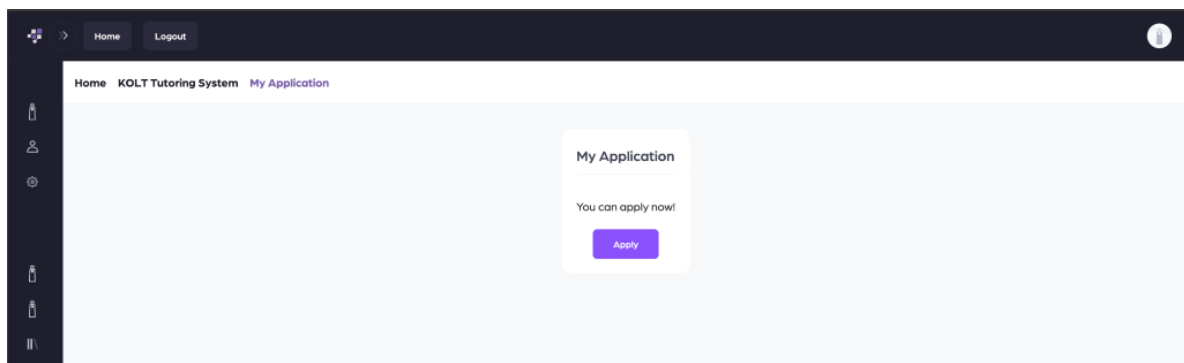


Figure 27 All students can apply to become a tutor when the applications are open.

When pressed, apply button redirects user to the application form. This form directly holds the user’s email address. This part is read only for the applicant. Applicant student can enter their name, GPA, courses they want to teach and periods that they are available weekly. Students who are already registered can not apply again.

The screenshot shows a web application interface for 'New Tutor Application'. The top navigation bar includes 'Home', 'KOLT Tutoring System', 'My Application', and 'Apply'. A sidebar on the left contains icons for user profile, application status, and other functions. The main form area is titled 'New Tutor Application' and contains the following fields:

- Name:** A text input field with the placeholder 'Enter Name'.
- Email:** A text input field containing 'serdogan18@ku.edu.tr'.
- GPA:** A text input field containing '0'.
- Courses:** A dropdown menu with the placeholder 'Select Courses'.
- Periods:** A dropdown menu with the placeholder 'Select Periods'.
- Weekly Hour:** A text input field containing '0'.

At the bottom of the form are two buttons: a purple 'Save' button and a light purple 'Cancel' button.

Figure 28 All students can apply to become a tutor when the applications are open.

After application, student can view the progress of their application. Status information is shown at top. Status can be “Pending”, “Approved” or “Denied”. Students can also edit and delete their application while the status is in pending state. After getting denied, or approved, no other application for that student is possible.

3.3.4.2 Cubicle Table

Cubicle table is an informative table for students listing the tutoring schedule information by mainly separating the teaching schedule plan by tutoring days.

After filtering the tutoring periods by their days, tutoring schedule is listed by cubicle and hour informations as columns and rows of the table. At each cubicle period conjunction cell, tutor name and the tutored course code information at that period and cubicle is printed.

Monday

Session Hours→ Cubicle No ↓	10:00 – 11:00	10:59 – 12:59	11:00 – 12:00
Cubicle 1	Hüseyin Emre Göksu ECON 333, PSYC 100		Sadık Muhammet Gencer BUSA 499, LAW 106
Cubicle 2	Naz King PSYC 100	Begüm Şen COMP 304	Deniz Yılmaz COMP 491, PSYC 100

Tuesday

Session Hours→ Cubicle No ↓	10:00 – 11:00	10:59 – 12:59	11:00 – 12:00
Cubicle 1	Ali Enes Gökçekuyu MGMT 506, MGMT 347		Zeynep Lara Kadioğlu INTL 201, INTL 301
Cubicle 2			Sadık Muhammet Gencer BUSA 499, LAW 106

Figure 28 All users can see weekly cubicle plans.

3.3.4.3 Schedule Table

Schedule table is an informative table for students listing the tutoring schedule information by mainly separating the teaching schedule plan by tutored courses.

After filtering the tutoring schedule by the courses, tutoring schedule is listed by day, time, place, tutor and cubicle number information. At each row respected information is shown.

This table's headers are set from the schedule text page of the staff. Table has link functionality at course codes and go to top buttons. Also whether or not showing the information of this table is set from the configuration page of staff pages.



Spring 2023 KOLT Tutoring Center Office Hours

ECON 333

Day	Time	Place	Tutor	Cubicle No
Monday	10:00-11:00	KOLT	Hüseyin Emre Göksu	1
Wednesday	10:00-11:00	KOLT	Hüseyin Emre Göksu	1
Thursday	12:00-13:00	KOLT	Hüseyin Emre Göksu	1

[Go to the top](#)

Figure 29 All users can see weekly schedule plans.

3.4 Problems Encountered

There exist several types of problems. We can list them both by chronologically and by their status.

Some of our problems start at the beginning of the project selection. After selecting KOLT Tutoring System project, we started working in synchronisation with both KOLT Center and IT. However, we did not have a direct relationship with KOLT Center. They send the IT a project requirement sheet however this sheet has some contradictions within. Whenever we had issues with this requirement sheet our advisors and mentors at IT made connection with KOLT. But this process required time and we ended up waiting in the first few weeks to get information about project requirements and get a clear vision in our heads.

After getting information from the KOLT, we started our project. But issues with requirements continued. We started by re-implementing a version of KOLT Tutoring System that

was provided us by IT. However this version was not in sync with the papers that we were also trying to satisfy. So we demanded a meeting with IT and KOLT. Turns out the application wanted from us was another application created on Mendix platform. All the plans changed and requirements started making sense weeks after project selection.

Our project required intense Database usage. However we were not provided with any information about the database type we were expected to use. We also waited for IT to give information about that. After getting the information that we needed to use PostgreSQL, we started creating entities and tables, but we were using separate databases and it was causing trouble on migrations, data creations and so on, so we needed to wait for each other whenever there was a migration or entity implementation to be in sync with each other and not causing conflicts. This also took our time but eventually we started using one common database, and the issue was solved.

Other problems were more in line with coding. One of the issues were that we didn't know how to separate view, model, entity and controller. So we just merged them in classes instead of separating. But after spending some time and learning it was ineffective we separated them clearly.

Another issue was about primary key, foreign key, null checkers and their constraints. We did not know which data was to be created initially, forced to be not null and since there were a lot of connections when tables were empty, there were lots of exceptions. We solved this problem by adding try-catch modules for checking null cases. And we solved the primary key foreign key issue about tutors by adding the first tutors id in code instead of automatic creation.

We also spend so much time on UI and matching the KUHUB style. Although there exists css files for many objects we still were not provided with some UI elements so they needed to be created by us. We later learnt some groups who work with IT were provided with at least the side bar or some UI elements, but we created many things from scratch.

Although we did solve most of our problems on project creation, we still have ongoing issues. One of the biggest problems of our application is its speed. The main reason we have a small applications because we work on local server. When we change this the issue about speed will be minimised. However, speed is also decreased because mass memory usage. Especially when we work with more than one database entity and do checks in for loops, we do take lots of data into memory. And since we store each data, it causes speed issues. Also we chose to use some functions that are slowing our system without knowing the results. We got this information after we did a final demo to our advisors at IT. If there will be a future work we can focus on solving these issues.

Also database entities after connection, not cascaded. So whenever we delete some connected entity we need to delete its connection and connected entity manually on code side. IT does not happen automatically on database side as intended. This issue is probably can be solved very easily but will take some time. We just need to pay attention to connection and their relations

on pages and carefully change the code.

4. Analysis and Results

As a result, we created a web application for the KOLT center in our university. Our application works as we want. We even think that it will be used in our university in the coming years. The main roles and their functions work the way we want. To explain in more detail: for example, people with the student role can see the weekly schedule and cubicle plans, and in addition, they can apply to become a tutor before the deadline. People with tutor role can see their tutoring information such as cubicle number, period, course information in addition to what the student has accessed. Persons with the head tutor role can see the information of the students and the course they are responsible for, apart from the pages that the student and tutor can access. The admin login, which is the biggest purpose of the project, also works correctly. In other words, the admin side can manage tutoring jobs very easily. Persons with the role of admin open the application to become a tutor to all students, and accept or reject them after the deadline. Moreover, the admin users can access the pages with all the courses, periods, cubicles and tutors, and they can add or remove data. The admin users can assign head tutors to courses, and assign cubicles to periods.

We held regular meetings with the KOLT office and IT, and determined their needs. The most important requirement IT wants is the UI side. They sent us documentation showing how each page, button and table should look like. These documents also specify the pages required for each role entry and the functions of the pages. The most important requirement KOLT wants from us is ease of use. They wanted the application to be easily managed by the admins and made it easy to use and understand. Consequently, we met those needs with the final version of the project. All roles perform the functions we want correctly. Both frontend and backend work properly. UI looks like IT wanted it to be. The database part is running smoothly and without conflicts and dangling. Both main tables and intermediate tables match each other and we can easily perform CRUD operations on these tables. Furthermore, the application is user friendly, and it is easy to use and understand it. Our project was very comprehensive, we made tests for each general page, we added try - catch blocks to the code. We have added error texts for each input entry based on the constraints such as null or uniqueness checker. In general, you can see the working version of the project from the demo videos in the appendix.

5. Conclusion

KOLT Tutoring System is a web application developed for Koc University Office of Learning and Teaching (KOLT) to be integrated with the new KUHub platform. The main goal of the project is to create a system to help the organization of tutoring services provided by KOLT. The most important features of our application are the application system and the calendar-like scheduling system. Moreover, UI of the application is specified by IT to be in accordance with KUHub. We included four different roles in our application, namely student, tutor, head tutor and staff. Each role has its own pages in the system. For example students can apply to become tutors, staff can view and approve or deny the application, tutors can view and edit their information, and head tutors can observe their assigned tutors. In the end, we successfully managed to implement all functionalities for each type of user. We developed UI of our application following the guidelines provided precisely in a responsive manner. Scheduling system and application system in our website works fine just as intended.

Possible improvements for the application may include adding an attendance system to be implemented for the tutor side. With the attendance data, staff can foresee the possible problems, plan and apply their future plans and investments. Moreover, since we worked locally, some algorithms work slower than intended. This problem can be solved by switching to a shared container such as Kubernetes and Docker.

6. References

- [1] "Key Figures." Koç University. [Online]. Available: <https://www.ku.edu.tr/en/explore/about/key-figures/>
- [2] "Kolt Tutoring Admin Panel." [Online]. Available: <https://kolttutoring.com/admin/>
- [3] KUHub. *Koç University*. [Online]. Available: <https://kuhub.ku.edu.tr/>
- [4] Google calendar. *Google Calendar* [Online]. Available: <https://www.steegle.com/google-products/google-calendar-faq>
- [5] Calendly. *The ultimate google calendar guide* [Online]. Available: <https://calendly.com/blog/google-calendar-tips#:~:text=Google%20Calendar%20is%20used%20by,features%20end%20up%20getting%20overlooked>
- [6] Doodle. *Start Scheduling Today With Doodle* [Online]. Available: <https://doodle.com/en/scheduling-software/#:~:text=Doodle%20is%20the%20best%20free,make%20a%20poll%20every%20month>

[7] Microsoft. *Microsoft Support* [Online]. Available : <https://support.microsoft.com/en-us/office/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8#:~:text=Calendar%20is%20the%20calendar%20and,Outlook%20Calendar%20and%20start%20typing>

7. Appendix

Demo Videos:

<https://drive.google.com/drive/folders/16jGbvPKxzm5NrI9RCxter4qmKKOhZqwd>

Diagram: <https://miro.com/app/board/uXjVME5ksrg=/>

GitHub (Final version in the Main branch): <https://github.com/KocUniversity-IT/STU-PRJ-23.KOLT-Tutoring-System>

Application Screenshots

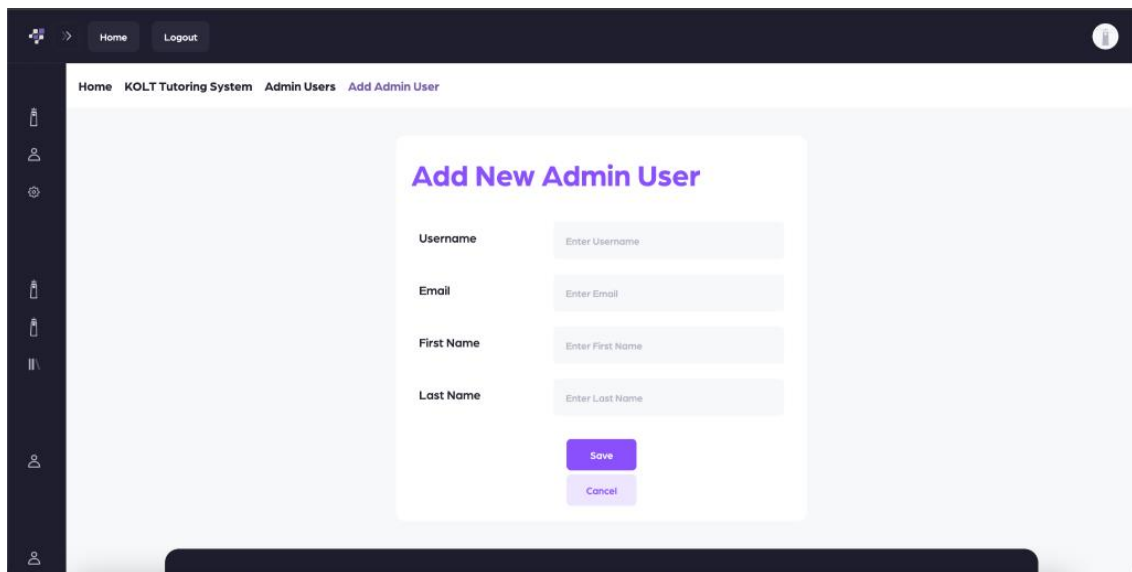
The screenshot shows a web application interface for the 'KOLT Tutoring System'. At the top, there is a dark navigation bar with 'Home' and 'Logout' buttons. Below this, a breadcrumb trail reads 'Home > KOLT Tutoring System > Admin Users > Add Admin User'. The main content area features a light blue sidebar with various icons. The central focus is a white modal box titled 'Add New Admin User' in purple. This modal contains four input fields: 'Username' (placeholder: 'Enter Username'), 'Email' (placeholder: 'Enter Email'), 'First Name' (placeholder: 'Enter First Name'), and 'Last Name' (placeholder: 'Enter Last Name'). At the bottom of the modal are two buttons: a purple 'Save' button and a light blue 'Cancel' button.

Figure 30 Admin users can add new admin user.

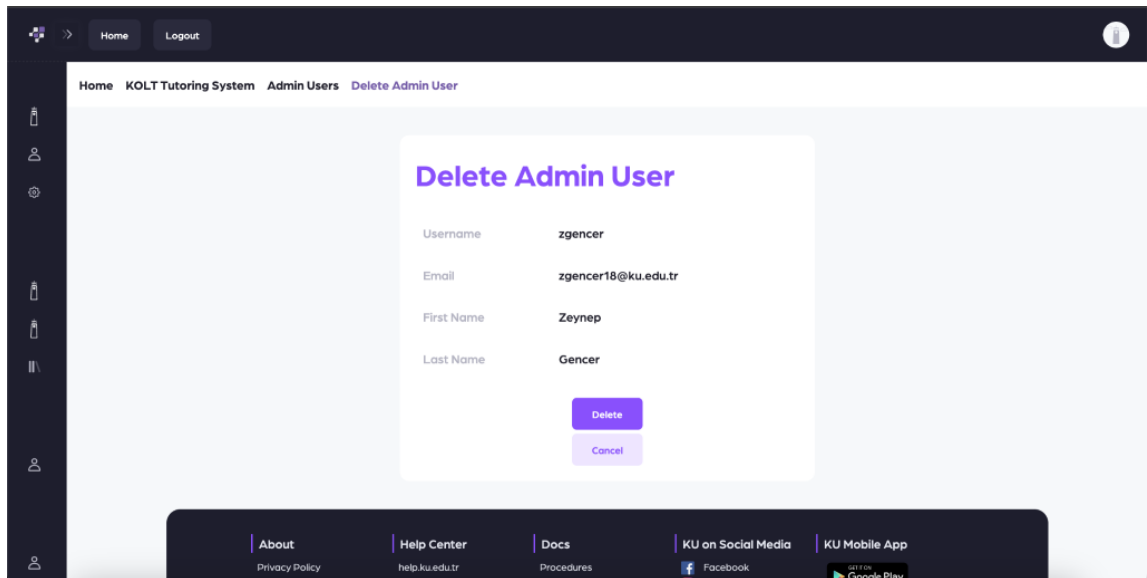


Figure 31 Admin users can delete an admin user.

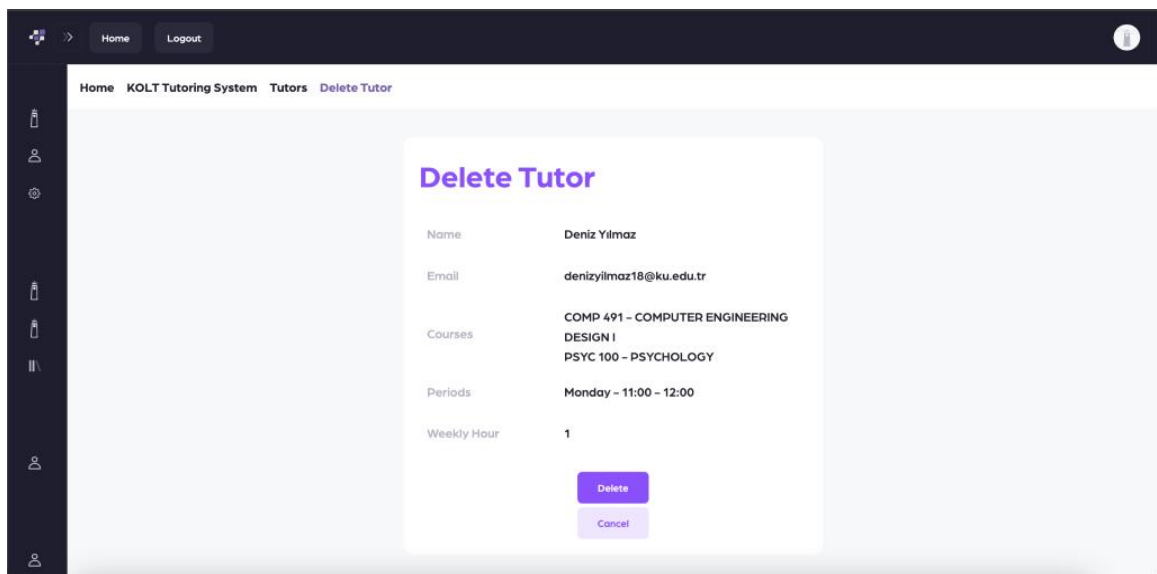


Figure 32 Admin users can delete a tutor from Tutors page and database.

Home KOLT Tutoring System Head Tutors Edit Head Tutor

Edit Head Tutor Assignment

Head Tutor Name: Zeynep Gencer

Head Tutor Email: zgencer18@ku.edu.tr

Head Tutor Course: PSYC 100

Tutors:

- Naz King, nking18@ku.edu.tr, PSYC 100
- Deniz Demirtürk, ddemirturk22@ku.edu.tr, MECH 201, PSYC 100
- Hüseyin Emre Gökusu, hgokusu18@ku.edu.tr, ECON 333, PSYC 100

Save Cancel

Figure 33 Admin users can edit an existing head tutor information.

Home KOLT Tutoring System Head Tutors Delete Head Tutor

Delete Head Tutor

Head Tutor Course: COMP 491

Head Tutor Name: Doğa Demirtürk

Tutors of Head Tutor: Deniz Yılmaz

Delete Cancel

About: Privacy Policy, FAQs
Help Center: help.ku.edu.tr, trackit.ku.edu.tr
Docs: Procedures
KU on Social Media: Facebook, Instagram, LinkedIn, Twitter
KU Mobile App: Get it on Google Play, Download on the App Store

Figure 34 Admin users can delete an existing head tutor form head tutor's list.

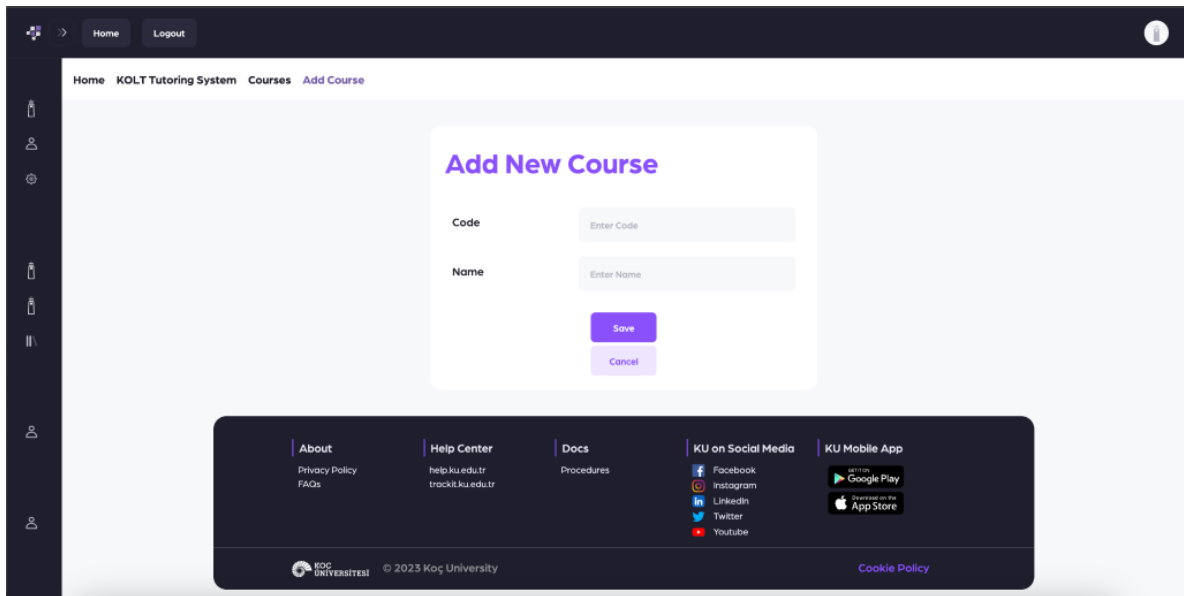


Figure 35 Admin users can add new course to Courses page and database.

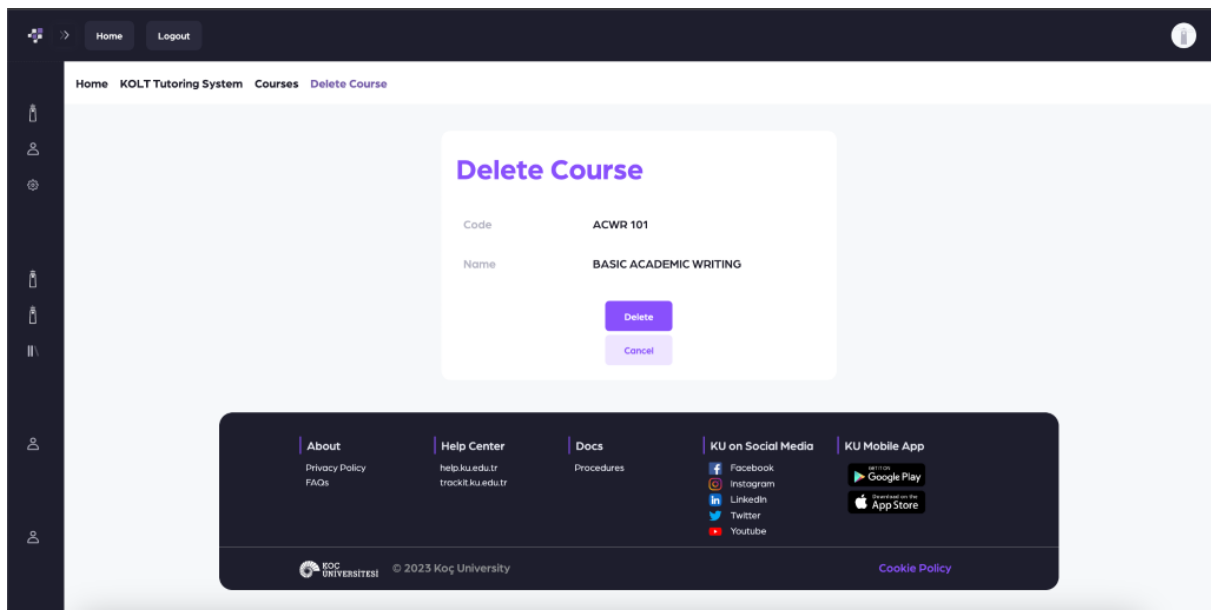


Figure 36 Admin users can delete a course from Courses page and database.

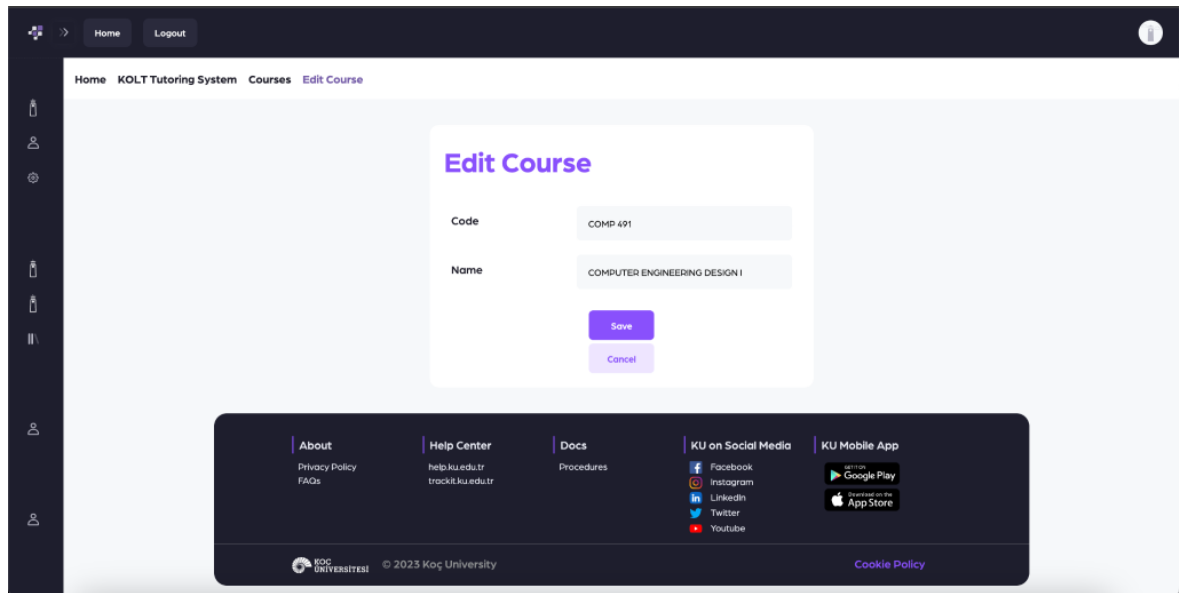


Figure 37 Admin users can edit an existing course.

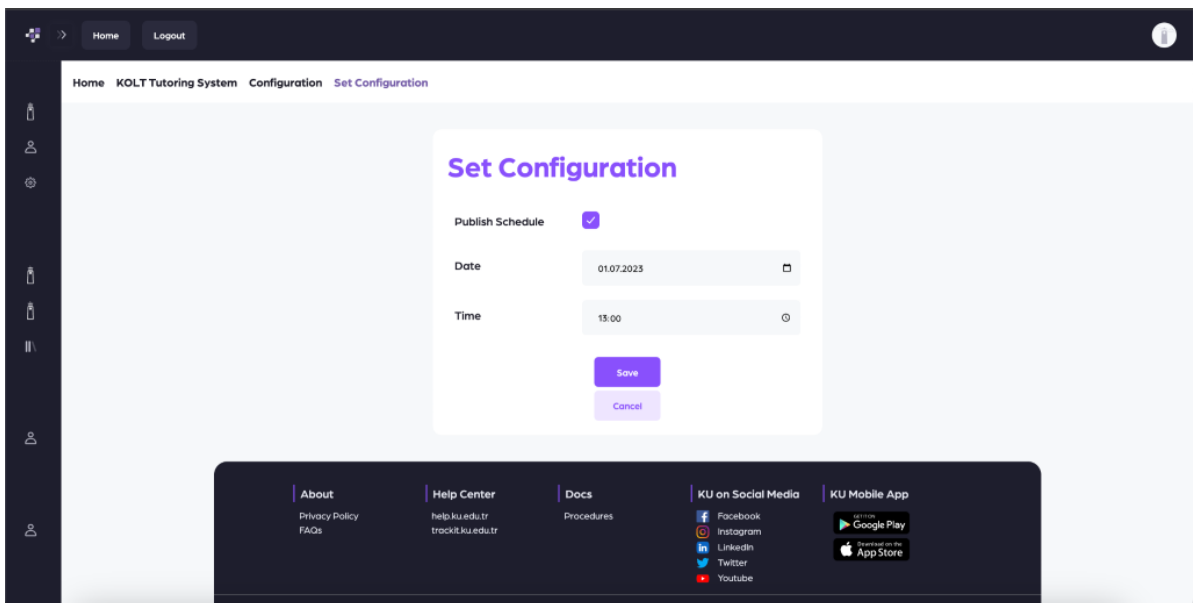


Figure 38 Admin users can set configuration.

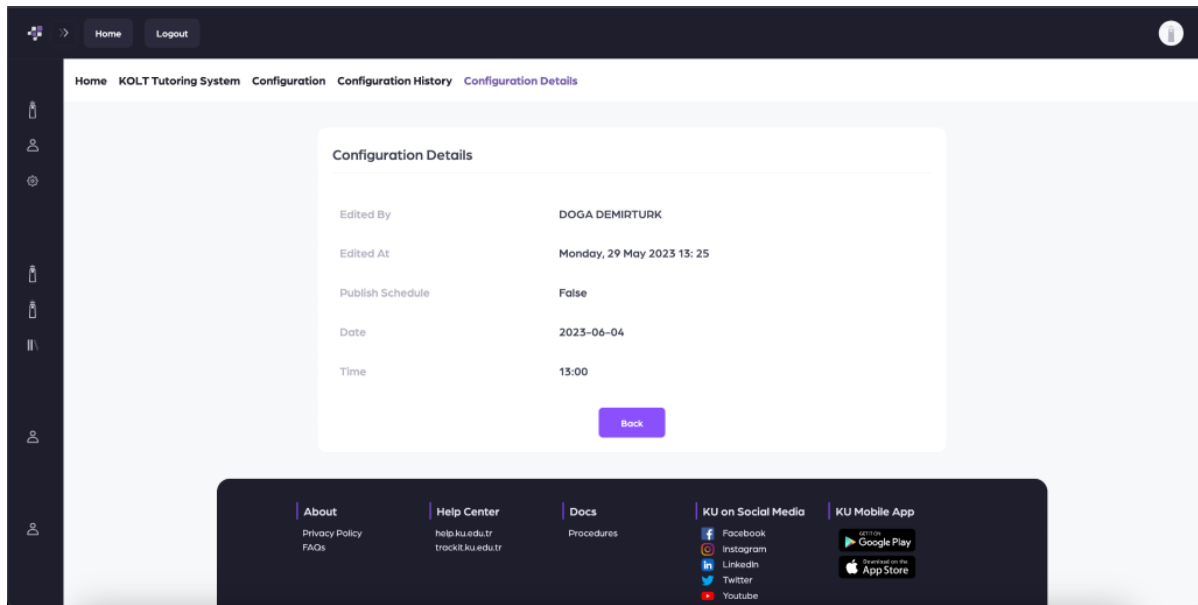


Figure 39 Admin users can see configuration details.

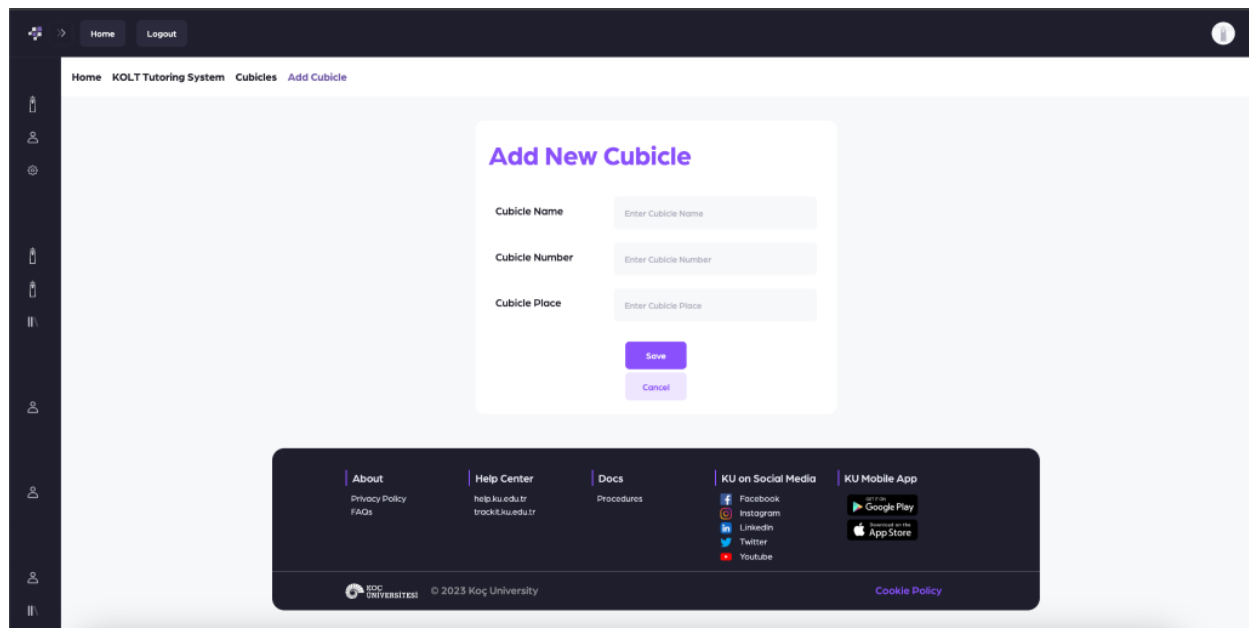


Figure 40 Admin users can add a new cubicle to cubicles table.

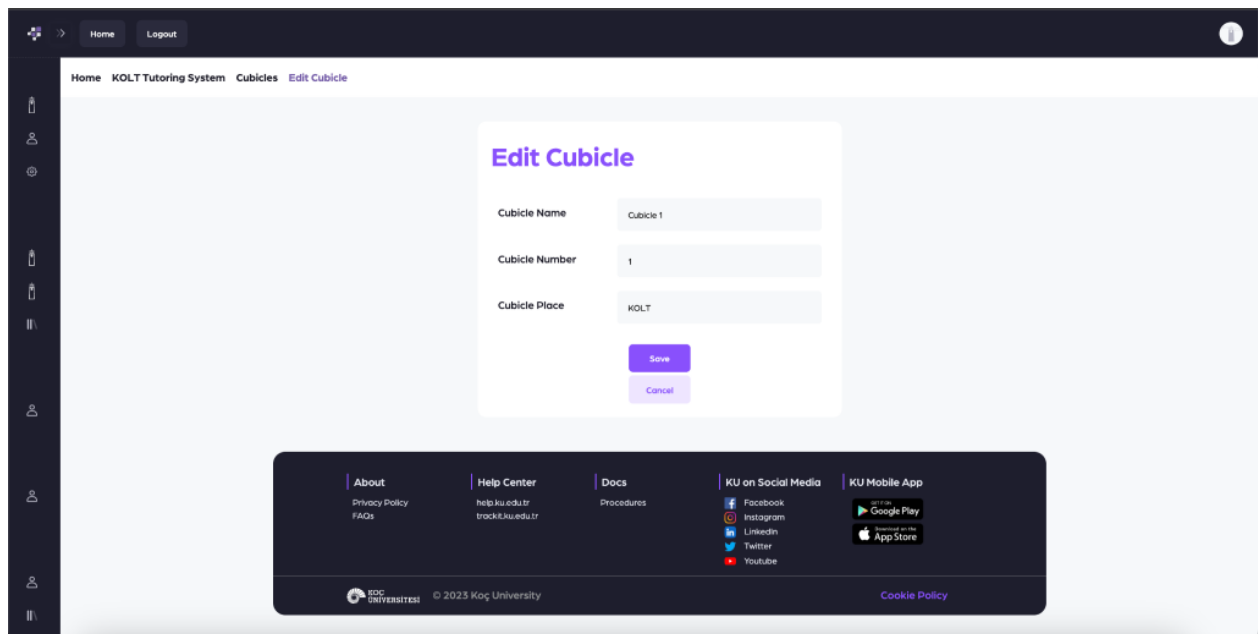


Figure 41 Admin users can edit an existing cubicle.

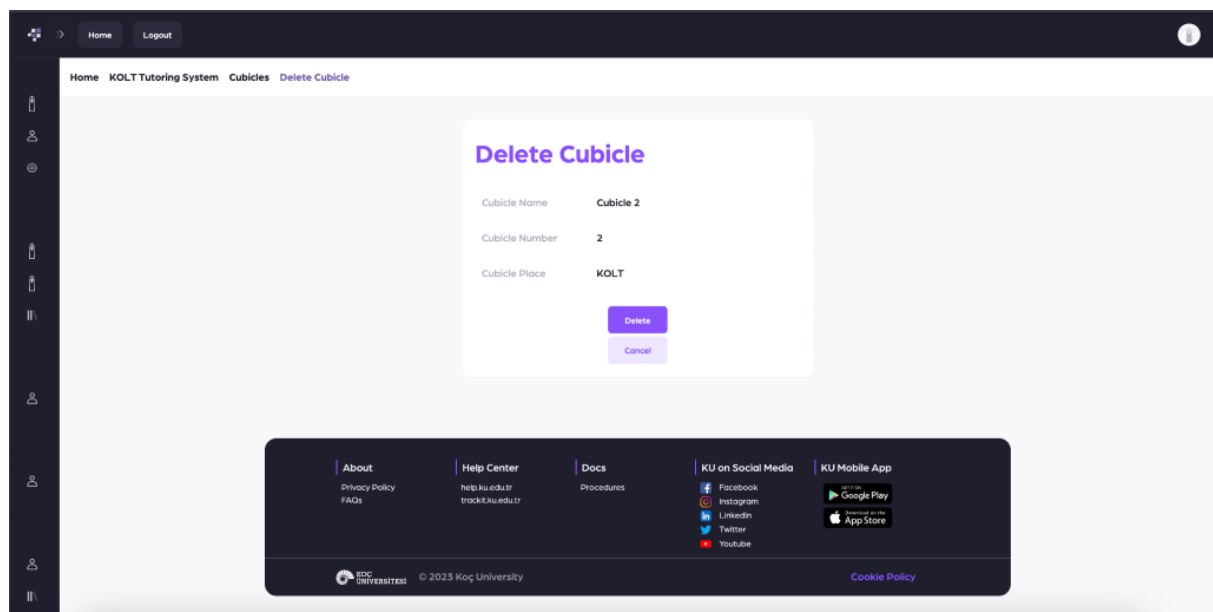


Figure 42 Admin users can delete a cubicle from cubicles table.

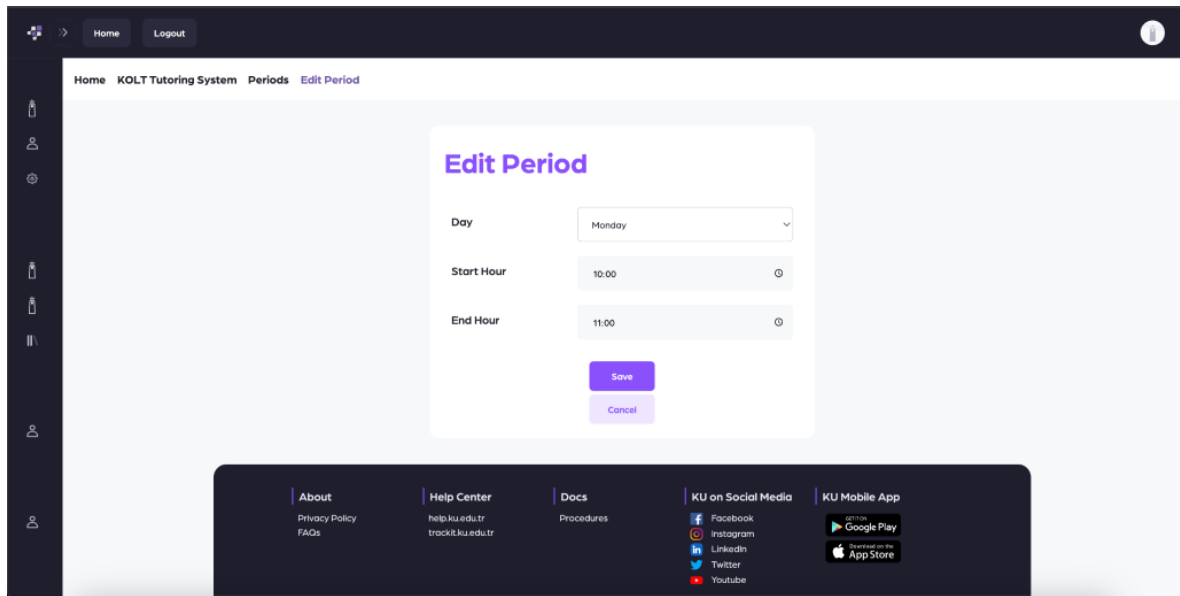


Figure 43 Admin users can edit an existing period.

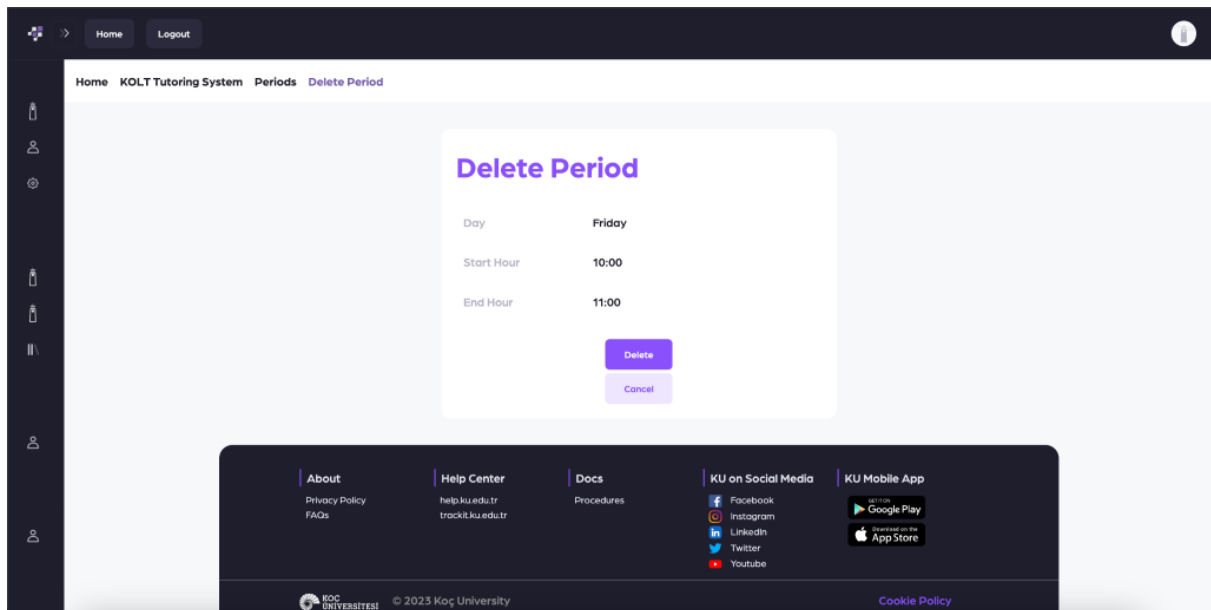


Figure 44 Admin users can delete a period from periods table.

Home KOLT Tutoring System Tutor Applications Edit Tutor Application

Edit Tutor Application

Name	Begüm Şen
Email	begumsen19@ku.edu.tr
GPA	3.5
Courses	COMP 301
Periods	Friday, 10:00-11:00
Weekly Hour	1

Save Cancel

Figure 45 Admin users can edit a tutor application.

Home KOLT Tutoring System Tutor Applications Delete Tutor Application

Delete Tutor Application

Name	Deniz Yılmaz
Email	denizyilmaz18@ku.edu.tr
Courses	PSYC 100
Periods	Tuesday, 10:00-11:00
Weekly Hour	1

Delete Cancel

About
Privacy Policy
FAQs

Help Center
help.ku.edu.tr
trackit.ku.edu.tr

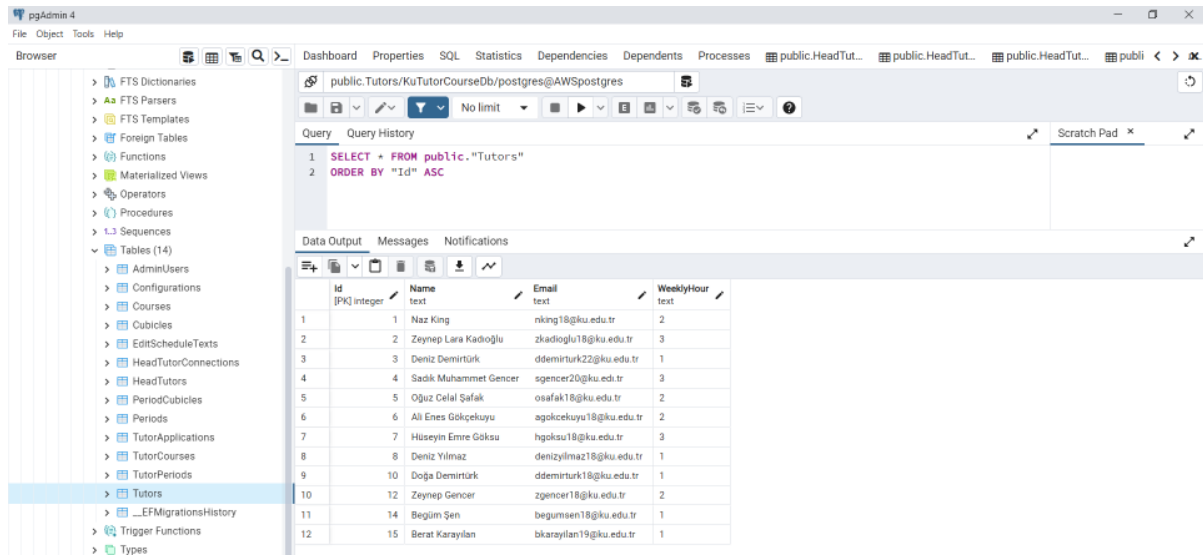
Docs
Procedures

KU on Social Media
Facebook
Instagram
LinkedIn

KU Mobile App
Google Play
https://kocuniku.ku.edu.tr/en/about-us/frequently-asked-questions/

Figure 46 Admin users can delete a tutor application.

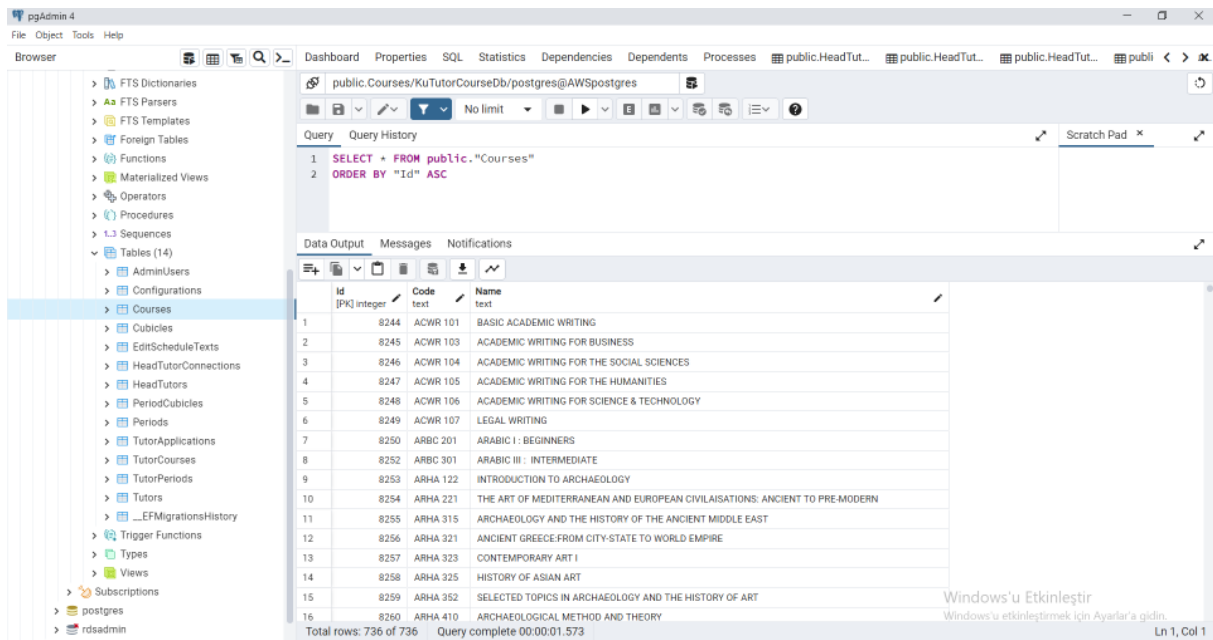
Database Screenshots



Query: `SELECT * FROM public."Tutors" ORDER BY "Id" ASC`

Id [PK] integer	Name text	Email text	WeeklyHour text
1	Naz King	nkking18@ku.edu.tr	2
2	Zeynep Lara Kadioğlu	zkadioglu18@ku.edu.tr	3
3	Deniz Demirtürk	ddemirturk22@ku.edu.tr	1
4	Sadık Muhammet Gencer	sgencer20@ku.edu.tr	3
5	Oğuz Celal Safak	osafak18@ku.edu.tr	2
6	Ali Enes Gökçekuyu	agokcekuyu18@ku.edu.tr	2
7	Hüseyin Emre Gökse	hgoksu18@ku.edu.tr	3
8	Deniz Yılmaz	denizyilmaz18@ku.edu.tr	1
9	Doğa Demirtürk	ddemirturk18@ku.edu.tr	1
10	Zeynep Gencer	zgencer18@ku.edu.tr	2
11	Begüm Şen	begumsen18@ku.edu.tr	1
12	Berat Karayilan	bkarayilan19@ku.edu.tr	1

Figure 47 Tutors table in the database.



Query: `SELECT * FROM public."Courses" ORDER BY "Id" ASC`

Id [PK] integer	Code text	Name text
1	8244 ACWR 101	BASIC ACADEMIC WRITING
2	8245 ACWR 103	ACADEMIC WRITING FOR BUSINESS
3	8246 ACWR 104	ACADEMIC WRITING FOR THE SOCIAL SCIENCES
4	8247 ACWR 105	ACADEMIC WRITING FOR THE HUMANITIES
5	8248 ACWR 106	ACADEMIC WRITING FOR SCIENCE & TECHNOLOGY
6	8249 ACWR 107	LEGAL WRITING
7	8250 ARBC 201	ARABIC I : BEGINNERS
8	8252 ARBC 301	ARABIC III : INTERMEDIATE
9	8253 ARHA 122	INTRODUCTION TO ARCHAEOLOGY
10	8254 ARHA 221	THE ART OF MEDITERRANEAN AND EUROPEAN CIVILISATIONS: ANCIENT TO PRE-MODERN
11	8255 ARHA 315	ARCHAEOLOGY AND THE HISTORY OF THE ANCIENT MIDDLE EAST
12	8256 ARHA 221	ANCIENT GREECE-FROM CITY-STATE TO WORLD EMPIRE
13	8257 ARHA 323	CONTEMPORARY ART I
14	8258 ARHA 325	HISTORY OF ASIAN ART
15	8259 ARHA 352	SELECTED TOPICS IN ARCHAEOLOGY AND THE HISTORY OF ART
16	8260 ARHA 410	ARCHAEOLOGICAL METHOD AND THEORY

Total rows: 736 of 736 Query complete 00:00:01.573

Figure 48 Courses table in the database.

pgAdmin 4

File Object Tools Help

Browser

- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1.3 Sequences
- ▼ Tables (14)
 - > AdminUsers
 - > Configurations
 - > Courses
 - > Cubicles
 - > EditScheduleTexts
 - > HeadTutorConnections
 - > HeadTutors
 - > PeriodCubicles
 - > Periods

public.Cubicles/KuTutorCourseDb/postgres@AWSpostgres

Query

```

1 SELECT * FROM public."Cubicles"
2 ORDER BY "Id" ASC

```

Data Output

	Id [PK] integer	CubicleName text	CubicleNumber text	CubiclePlace text
1	16	Cubicle 2	2	KOLT
2	17	Cubicle 1	1	KOLT

Figure 49 Cubicles table in the database.

pgAdmin 4

File Object Tools Help

Browser

- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1.3 Sequences
- ▼ Tables (14)
 - > AdminUsers
 - > Configurations
 - > Courses
 - > Cubicles
 - > EditScheduleTexts
 - > HeadTutorConnections
 - > HeadTutors
 - > PeriodCubicles

public.AdminUsers/KuTutorCourseDb/postgres@AWSpostgres

Query

```

1 SELECT * FROM public."AdminUsers"
2 ORDER BY "Id" ASC

```

Data Output

	Id [PK] integer	Username text	Email text	FirstName text	LastName text
1	9	dogadm	ddemirturk18@ku.edu.tr	Doga	Demirturk
2	10	sinancemerdogan	serdogan18@ku.edu.tr	Sinan Cem	Erdoğan
3	11	zgencer	zgencer18@ku.edu.tr	Zeynep	Gencer
4	12	dylmaz	dylmaz18@ku.edu.tr	Deniz	Yilmaz

Figure 50 AdminUsers table in the database.

pgAdmin 4

File Object Tools Help

Browser

public.Periods/KuTutorCourseDb/postgres@AWSpostgres

Query

```
1 SELECT * FROM public."Periods"
2 ORDER BY "Id" ASC
```

Data Output

Id [PK] integer	Day text	StartHour text	EndHour text
1	58	Monday	10:00
2	59	Tuesday	10:00
3	60	Wednesday	10:00
4	61	Thursday	12:00
5	62	Friday	10:00
6	63	Monday	11:00
7	64	Tuesday	11:00
8	65	Wednesday	11:00
9	66	Thursday	11:00
10	67	Friday	11:00
11	68	Wednesday	10:58

Figure 51 Periods table in the database.

pgAdmin 4

File Object Tools Help

Browser

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (14)

AdminUsers

Configurations

Courses

Cubicles

EditScheduleTexts

HeadTutorConnections

HeadTutors

PeriodCubicles

Periods

TutorApplications

TutorCourses

Dashboard Properties SQL Statistics Dependencies Dependents Processes

public.HeadTut... public.HeadTut... public.HeadTut...

public.TutorApplications/KuTutorCourseDb/postgres@AWSpost...

No limit

Query Query History

1 SELECT * FROM public."TutorApplications"

2 ORDER BY "Id" ASC

Data Output Messages Notifications

	<div>Id [PK] integer</div>	<div>Name text</div>	<div>Email text</div>	<div>GPA double precision</div>	<div>CourseId integer[]</div>	<div>PeriodId integer[]</div>	<div>WeeklyHour integer</div>	<div>Status text</div>
1	161	Defne Ersavaş	dersavas18@ku.edu.tr	3.5	(8242)	{60}	1	Denied
2	162	Deniz Yılmaz	denizyilmaz18@ku.edu.tr	3.67	(8886)	{59}	1	Approved
3	184	Begüm Şen	begumsen19@ku.edu.tr	3.5	(8346)	{62}	1	Pending
4	186	Sinan Cem Erdoğan	serdogan18@ku.edu.tr	3.19	(8244)	{58}	1	Pending

Figure 52 TutorApplications table in the database.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (14)' folder is expanded, and 'TutorCourses' is selected. The main pane displays the 'Data Output' tab for the 'public.TutorCourses/KuTutorCourseDb/postgres@AWSpostgres' connection. A query is executed, showing the following data:

TutorId [PK] integer	CourseId [PK] integer
1	8886
2	8578
3	8581
4	8744
5	8886
6	8288
7	8619
8	8466
9	8784

Figure 53 TutorCourses table in the database.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (14)' folder is expanded, and 'TutorPeriods' is selected. The main pane displays the 'Data Output' tab for the 'public.TutorPeriods/KuTutorCourseDb/postgres@AWSpostgres' connection. A query is executed, showing the following data:

TutorId [PK] integer	PeriodId [PK] integer	PeriodTutorId integer
1	58	257
2	73	313
3	60	307
4	62	308
5	64	302
6	60	299
7	63	262
8	64	277
9	67	263
10	65	264
11	66	265

Figure 54 TutorPeriods table in the database.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (14)' folder is expanded, and 'PeriodCubicles' is selected. The main pane displays a SQL query: `SELECT * FROM public."PeriodCubicles" ORDER BY "PeriodCubicleId" ASC`. The 'Data Output' tab shows the results of the query in a table format.

	PeriodCubicleId [PK] integer	PeriodTutorId integer	CubicleId integer
1	513	257	16
2	519	263	17
3	520	264	16
4	522	266	17
5	532	276	17
6	546	290	17
7	556	300	17
8	557	277	16
9	558	299	16
10	559	265	17
11	560	296	16
12	562	302	17

Figure 55 PeriodCubicles table in the database.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Tables (14)' folder is expanded, and 'HeadTutors' is selected. The main pane displays a SQL query: `SELECT * FROM public."HeadTutors" ORDER BY "Id" ASC`. The 'Data Output' tab shows the results of the query in a table format.

	Id [PK] integer	HeadTutorsTutorId integer	TutorId integer[]	CourseId integer
1	57	10	{8,9}	8360
2	59	12	{1,3,7}	8886

Figure 56 HeadTutors table in the database.

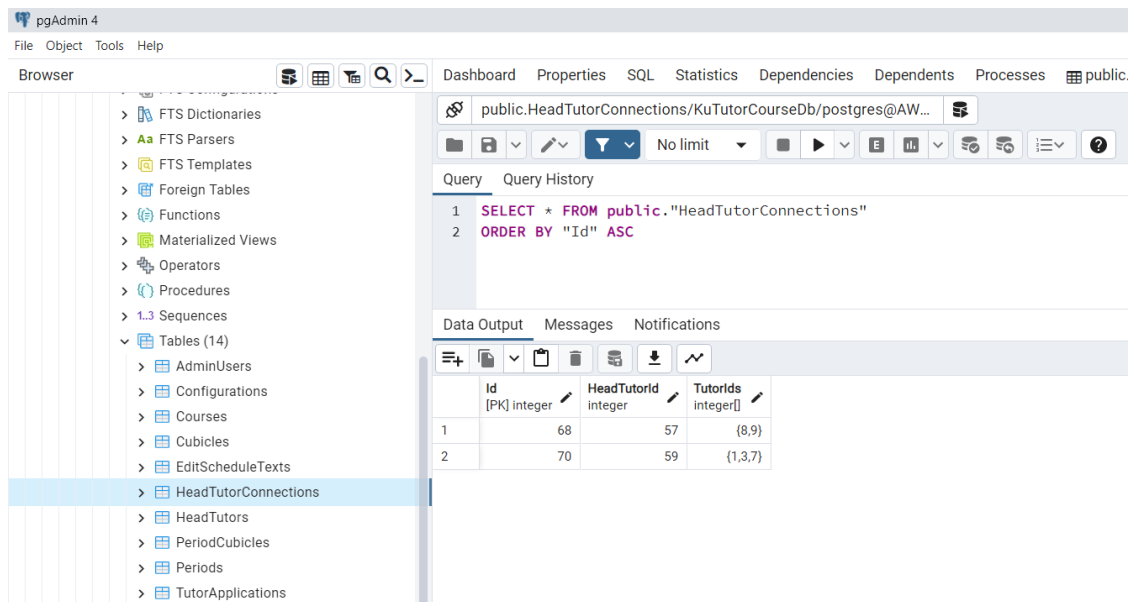


Figure 57 HeadTutorConnections table in the database.