This project aims to manage graduation project process of students at one center.

Application Requirements:

- There are three main entities: Announcements,homeworks,projects
- Announcements contain following attributes: Header (string) ,detail (string), files (array of strings contain filenames.)
- Homeworks contain a last submission date and time attributes in addition to attributes of announcements.
- Projects contain student number, project title and project advisor. All of them are string.
- Users are divided into two groups: Students and instructors.
- Students may show only their own projects and all announcements and homeworks.
- Instructors may show all projects,announcements and homeworks.
- A student may have at most one project
- Only instructors are allowed to insert,modify and delete anything.
- In this source code, three dummy users are created. First one is instructor with username “a” and password “b”. Second one is student with username=“c” and password=“d”. Third one is also a student with username=“e” and password=“f”, but it does not take any graduation projects and does not allowed to enter the system.
- Only students can upload homeworks.
- Students may download only their own homeworks. Instructors can download all homeworks.

Used Technologies

1. **Backend Technologies**

- Node.js: A JavaScript server language.
- Express.js: A minimal and flexible Node.js web application framework
- MongoDB: A NoSQL database system that uses JSON objects.
- Multer: A Node.js library that used for file uploadings.
- Fs: A Node.js file system module.
- Fs-extra: A Node.js file system module which is used for directory removing and moving.
- Body-parser: A library helps converting POST requests into the request body.

2. **Frontend Technologies**

- Pug.js: It is the most popular template engine works with node.js. It uses standard JavaScript syntax and a modified version of HTML syntax.
- Bootstrap: It is a CSS framework which is used for building responsive, fluent and elegant design
- Font Awesome: This is an useful icon library.
- jQuery: A JavaScript library which simplifies JavaScript programming

Technical Documentation

1. **Database Creation**

   Database initialization is performed in “dbinit.py” file.

   1. Import MongoDB library.
   2. Define database URL.
   3. Create a MongoDB client.
   4. Create a database with name of “db”

5. Create “announcements”,”homeworks” and “projects” collections (Collections are equivalent of “tables” in SQL.

2. Database Dropping

Database initialization is performed in “dbdrop.py” file.

1. Import MongoDB library.
2. Define database URL.
3. Create a MongoDB client.
4. Drop the database with name of “db”
5. Destroy “announcements”,”homeworks” and “projects” collections (The term of “collection” is equivalent of “table” in SQL.

3. Preparing Server

This operation and remaining backend operations is defined at “server.js” file.

1. Import HTTP library.
2. Import Express.js library
3. Create an Express router with name of “app”
4. Import Pug.js library
5. Import Multer library
6. Import fs library.
7. Import fs-extra library.
8. Create changed variable which keeps whether a project is successfully added or updated. Default value is true.
9. Import “languages.js” file contains Turkish and English texts of website as an array of JSON objects.
10. Create language value to keep language of pages. Default value is English
11. Create a Multer upload object. Destination directory is directory of server file
12. Import body-parser library.
13. Create a server that uses “app” application and listens port 2020.
14. Import express-session library.
15. Import MongoDB library.
16. Create a MongoDB client
17. Define database URL
18. Use body-parser in this application
19. Match “/static”,”/announcements” and “/homeworks” routes with corresponding folders.
20. Create a session with a secret key and use it in this application.
21. Allow CORS requests.
22. Set Pug.js as default HTML template engine.
23. Set “static” subfolder of project folder as default folder of Pug files.

4. Loading Login File

“/ login” route GET request.

If user is authenticated, redirect the user to the main page. Else, load the login.pug file with username,password,language and website texts as parameters.

5. Authentication

"/ login" route POST request.

When user submitted its ITU Username and Password, get this

Register username and password information to session

If requested data are valid. If valid, redirect to the main page. Otherwise, go back to the login page.

6. Loading Main Page

Default route GET request.

When user enters the system or successfully authenticated, get this

If user successfully authenticated, check if user is a student.

Connect to the database

Set "changed" to its default value.

If user is student, find its project. Result is a JSON object.

For every user, search all documents from announcements and homeworks collections. Result is an array of JSON objects.

Create an empty array with name of "file_found".

If user is student, check every homework if current user uploaded it and push "true" or "false" values to the array regarding with result.

If user is instructor, check every homework if at least one user uploaded it and push "true" or "false" values to the array regarding with result.

Load "main.pug" file. If user is a student, pass its project object as parameter. If user is an instructor, pass "changed" object explained above as parameter. For every user, pass announcements, homeworks and file_found arrays,current username,language and website texts additional parameters.

Close the database

If user enters the system first time, redirect to login page.

7. Inserting Documents

Default route POST request.

If a user attempts to insert a new announcement, homework or project, get this.

Connect to the database.

If request contains an advisor, inserting document is a project.

If inserting document is not a project and the request does not contain a date, inserting document is an announcement. Otherwise, it is homework.

In Multer, request files are uploaded to folder of the project with a different name. "name" attribute of uploaded file object keeps changed name and "originalname" attribute keeps the original name.

If inserting document is not a project, take all files in request and push their names into an array. Then, create an object which contains files array and other request values and add it to the database. In MongoDB, every document has a 12-byte length unique object ID. At last, find ID of inserted object and move every request file to the "/announcements/"+ObjectID or "/homeworks/"+ObjectID folders with their original name.

If inserting document is a project, create an object with request values. If given student has already a project, don't insert anything and set "changed" variable to false. Changed variable is to show error message if inserting or updating a project is successful. Otherwise, insert project to the collection.

Redirect to the main page

8. **Search Projects**
   "/searchProjects" route POST request.
   When a user attempts to search projects, get this.
   Search projects fits to given query and return results.
   This operation does not perform any reloading.
9. **Update a Document**
   '/update' route POST request.
   "type" parameter of the route determines what kind of document will be updated.
   Convert requested ID to a MongoDB ObjectId.
   Connect to the database.
   Find the object with this ID.
   If "type" parameter is not "projects", take all files in request and push their original names into an array. Then, move request files to the "/announcements/"+ObjectID or "/homeworks/"+ObjectID folders with their original name If they.are not existed in the document before.
   If document type is "homeworks" and any student upload exists, change "/homeworks/uploads/"+Old_homework_header folder to the "/homeworks/uploads/"+New_homework_header
   Create an object with request values. and make update.
   Redirect to the main page.
10. **Get ID of Index**
    '/getIDOfIndex' route POST request.
    "type" parameter of the route determines what kind of document will be found.
    This route finds all documents of related collection and returns ID of document in requested index
    This operation does not perform any reloading.
11. **Delete a Document**
    '/delete' route POST request.
    "type" parameter of the route determines what kind of document will be deleted.
    Connect to the database
    If type parameter is projects, convert requested ID to a MongoDB ObjectId and find project with this ID. Otherwise, find all documents in that type.
    If "type" parameter is not "projects",find document with requested index. Then,delete the folder "/announcements/"+ObjectID or "/homeworks/"+ObjectID.
    If document type is "homeworks" and any student upload exists, delete "/homeworks/uploads/"+Old_homework_header folder

Redirect to the main page.

12. **Delete All Documents**
    '/deleteall' route POST request.
    "type" parameter of the route determines what kind of documents will be completely deleted.
    Connect to database and delete all documents in selected collection.
    Delete '/announcements' or '/homeworks' folder, if type is not "projects".
13. **Delete Any Source File (Only Available For Announcements And Homeworks)**
    '/deletefile' route POST request.
    "type" parameter of the route determines deleted file is in which type of a document.
    Connect the database and update the document such that remove the file string at requested index of files array. Then delete the file with this name in folder "/announcements/" +ObjectID or "/homeworks/"+ObjectID.
14. **Delete All Source Files of a collection (Only Available For Announcements And Homeworks)**
    '/deleteallfiles' route POST request.
    "type" parameter of the route determines deleted files are in which type of a document.
    Connect the database and update the document such that clear files array. Then delete folder "/announcements/" +ObjectID or "/homeworks/"+ObjectID.
15. **Upload a homework**
    '/uploadhomework' route POST request.
    When a student uploads a homework, previously uploaded homework is deleted. Homeworks are moved to '"/homeworks/uploads/"+[Homework Header]+'/'+[Current Username]' folder.
16. **Download Uploaded Homework Files**
    '/download'+homework_header+'.zip' route GET request
    OS library is used to create a temporary directory and zip-local library is used for compression.
    If the user is instructor, download homeworks of all users.
    (/homeworks/uploads/"+[Homework Header])
    If the user is a student, download homework of this user.
    (/homeworks/uploads/"+[Homework Header]+'/'+[Current Username])
    To download, create a zip file in a temporary folder and send it.
17. **Change Language**
    '/changelanguage' route GET request.
    When user clicks on a language (English or Turkish), language is changed and user is redirected to login page.
18. **Log out**
    '/logout' route GET request.
    When a user clicks on logout icon, session is terminated and user is redirected to login page.

Use Cases:

Use Case #1 – Logging In	
Primary Actor:	ITU Computer Engineering students and instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students and instructors can enter the system by providing their ITU username and password

Basic Flow:	**1.** User directed to login page **2.** ITU username and password are supplied by user and submitted **3.** System validates the credentials and redirects user to the main page
Alternative Flow:	**1.** User directed to login page **2.** ITU username and password supplied by user and submitted **3.** If credentials are invalid or the student does not take any graduation project course, system shows the related error message and redirects to the login page again

Use Case #2 – Show all announcements	
Primary Actor:	ITU Computer Engineering students and instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students and instructors can show all announcements
Basic Flow:	**1.** User is logged in and redirected to the main page **2.** In main page, system shows all announcement headers and available operations
Alternative Flow:	**1.** User is logged in and redirected to the main page **2.** There are no announcements so “No announcements found” message is shown by the system

Use Case #3 – Add an announcement	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can add announcements
Basic Flow:	**1.** Announcement add form is displayed by the system **2.** Header, details and attachment files information of the announcement are provided by the user **3.** System adds announcement to the database, upload attachment files and redirects user to the main page
Alternative Flow:	**1.** Announcement add form is displayed by the system **2.** User doesn’t provide a header so announcement is not added

Use Case #4 – Update an announcement	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal

Brief:	ITU Computer Engineering instructors can update announcements
Basic Flow:	**1.** User clicks on “Update” button in “Operations” column of any announcement **2.** Announcement update form is displayed by the system **3.** New header and new details information of the announcement are provided by the user **4.** User may add new attachment files. **5.** System updates related announcement to the database, upload new attachment files if requested and redirects user to the main page
Alternative Flow:	**1.** User clicks on “Update” button in “Operations” column of any announcement **2.** Announcement update form is displayed by the system **3.** New header of the announcement is empty so announcement is not updated or **1.** There are no announcements so user can not any update any announcement

Use Case #5 – Delete an announcement	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete announcements
Basic Flow:	**1.** User clicks on “Delete” button in “Operations” column of any announcement **2.** System deletes related announcement from the database, deletes attachment files of the related announcement and redirects user to the main page
Alternative Flow:	**1.** There are no announcements so user can not delete any announcement

Use Case #6 – Delete all announcements	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete all announcements
Basic Flow:	**1.** User clicks on “Delete All” button on the bottom of the announcements table **2.** System deletes all announcements from the database, deletes all attachment files of announcements and redirects user to the main page
Alternative Flow:	**1.** There are no announcements so user can not delete all announcements

Use Case #7 – Show announcement details	
Primary Actor:	ITU Computer Engineering students and instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students and instructors can show announcement details
Basic Flow:	**1.** User clicks on “Show Details” button in “Operations” column of any announcement **2.** Announcement details and attachment files are displayed by the system
Alternative Flow:	**1.** There are no announcements so user can not show announcement details

Use Case #8 – Delete an attachment file of an announcement	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete an attachment file of an announcement
Basic Flow:	**1.** User clicks on “Show Details” button of any announcement. **2.** System shows attachment files of the announcement and every file has its own “Delete” button. **3.** User clicks on any of these buttons **4.** System deletes the file redirects user to the main page.
Alternative Flow:	**1.** There are no announcements so user can not delete any file of any announcement

Use Case #9 – Delete all attachment files of an announcement	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete all attachment files of an announcement
Basic Flow:	**1.** User clicks on “Show Details” button of any announcement. **2.** System shows “Delete All Files” button at bottom of the announcement attachment files **3.** User clicks on this button **4.** System deletes all attachment files of the announcement and redirects user to the main page.
Alternative Flow:	**1.** There are no announcements so user can not delete all files of any announcement

Use Case #10 – Show all homeworks	
Primary Actor:	ITU Computer Engineering students and instructors

Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students and instructors can show all homeworks
Basic Flow:	**1.** User is logged in and redirected to the main page **2.** In main page, system shows all homework headers, last submission dates and available operations
Alternative Flow:	**1.** User is logged in and redirected to the main page **2.** There are no homeworks so "No homeworks found" message is shown by the system

Use Case #11 – Add a homework	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can add homeworks
Basic Flow:	**1.** Homework add form is displayed by the system **2.** Header, details, source files and last submission date information of the homework are provided by the user **3.** System adds homework to the database, upload source files and redirects user to the main page
Alternative Flow:	**1.** Homework add form is displayed by the system **2.** User doesn't provide a header so homework is not added

Use Case #12 – Update a homework	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can update homeworks
Basic Flow:	**1.** User clicks on "Update" button in "Operations" column of any homework **2. H**omework update form is displayed by the system **3.** New header, new details and new last submission date of the homework are provided by the user **4.** User may add new source files. **5.** System updates related homework to the database, upload new source files if requested and redirects user to the main page
Alternative Flow:	**1.** User clicks on "Update" button in "Operations" column of any homework **2.** Homework update form is displayed by the system

	3. New header of the homework is empty so homework is not updated or **1.** There are no homeworks so user can not any update any homework

Use Case #13 – Delete a homework	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete homeworks
Basic Flow:	**1.** User clicks on “Delete” button in “Operations” column of any homework **2.** System deletes related homework from the database, deletes source files and student uploads of the related homework and redirects user to the main page
Alternative Flow:	**1.** There are no homeworks so user can not delete any homework

Use Case #14 – Delete all homeworks	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete all homeworks
Basic Flow:	**1.** User clicks on “Delete All” button on the bottom of the a homeworks table **2.** System deletes all homeworks from the database, deletes all source files and student uploads of homeworks and redirects user to the main page
Alternative Flow:	**1.** There are no homeworks so user can not delete all homeworks

Use Case #15 – Show homework details	
Primary Actor:	ITU Computer Engineering students and instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students and instructors can show homework details
Basic Flow:	**1.** User clicks on “Show Details” button in “Operations” column of any homework **2.** Homework details and source files are displayed by the system
Alternative Flow:	**1.** There are no homeworks so user can not show homework details

Use Case #16 – Delete a source file of a homework	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete a source file of a homework
Basic Flow:	**1.** User clicks on “Show Details” button of any homework **2.** System shows attachment files of the homework and every file has its own “Delete” button. **3.** User clicks on any of these buttons **4.** System deletes the file redirects user to the main page.
Alternative Flow:	**1.** There are no announcements so user can not delete any file of any announcement

Use Case #17 – Delete all source files of a homework	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete all source files of a homework
Basic Flow:	**1.** User clicks on “Show Details” button of any homework **2.** System shows “Delete All Files” button at bottom of the homework source files **3.** User clicks on this button **4.** System deletes all source files of the homework and redirects user to the main page.
Alternative Flow:	**1.** There are no homeworks so user can not delete all files of any homework

Use Case #18 – Upload homework	
Primary Actor:	ITU Computer Engineering students
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students can upload their homeworks
Basic Flow:	**1.** User selects homework files and click on “Upload homework” button of the related homework **2.** System uploads homework files
Alternative Flow:	**1.** If submission time is expired, user can not upload the homework or **1.** If user sends a new homework, system deletes old homework files and uploads new homework files

Use Case #19 – Download Uploaded Homework Files	
Primary Actor:	ITU Computer Engineering students and instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering students can download their homeworks
Basic Flow:	**1.** User clicks on “Download Uploaded Homework Files” link of any homework **2.** If user is student**, s**ystem sends homework files of the student as a ZIP file **3.** If user is an instructor**, s**ystem sends all homework files as a ZIP file

Use Case #20 – Add a project	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can add projects
Basic Flow:	**1.** Announcement add form is displayed by the system **2.** Student,title and advisor information of the project are provided by the user **3.** System adds project to the database, upload project files and redirects user to the main page
Alternative Flow:	**1.** Project add form is displayed by the system **2.** User doesn’t provide a student,title or advisor so project is not added

Use Case #21 – Update a project	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can update projects
Basic Flow:	**1.** User clicks on “Update” button in “Operations” column of any project **2.** Project update form is displayed by the system **3.** New student,new advisor and new title information of the project are provided by the user **5.** System updates related project to the database and redirects user to the main page
Alternative Flow:	**1.** User clicks on “Update” button in “Operations” column of any project **2.** Project update form is displayed by the system **3.** New student,title or advisor of the project is empty so project is not updated or

	1. User clicks on “Update” button of any project **2.** Project update form is displayed by the system **3.** New student has already a project in the system so project is not updated. or **1.** There are no projects so user can not any update any project

Use Case #22 – Delete a project	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete projects
Basic Flow:	**1.** User clicks on “Delete” button in “Operations” column of any project **2.** System deletes related project from the database and redirects user to the main page
Alternative Flow:	**1.** There are no projects so user can not delete any project

Use Case #23 – Delete all projects	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can delete all projects
Basic Flow:	**1.** User clicks on “Delete All” button on the bottom of the projects table **2.** System deletes all projects from the database and redirects user to the main page
Alternative Flow:	**1.** There are no projects so user can not delete all projects

Use Case #24 – Search projects	
Primary Actor:	ITU Computer Engineering instructors
Scope:	ITU Computer Engineering Student Graduation Projects Automation System
Level:	User goal
Brief:	ITU Computer Engineering instructors can search from projects
Basic Flow:	**1.** User selects search criteria, enters search text,clicks “Search” button **2.** System returns projects which contains given text in given search criteria