maltepe university
istanbul
www.maltepe.edu.tr

SE 307 – Database Management Systems

Term Project
First Part – 2024/25 Fall

Assoc. Prof. Dr. Volkan TUNALI

Prepared by:
21 07 06 017 - Muhammed Numan Karaca
21 07 06 028 - Ömer Faruk Özer
21 07 06 039 - Berke Ensel

Revision History

Date	Version	Author	Change
20/11/2024	1.0	All Team	File created and project details searched.
23/11/2024	1.1	All Team	Created tables and testing.
08/12/2024	1.2	All Team	Fix codes and last version of first part.
20/12/2024	2.0	All Team	Reading Documentation and Research
27/12/2024	2.1	All Team	Added Frontend - Backend
03/01/2025	2.3	All Team	Some fixes and Create Documentation

Table of Contents

REVISION HISTORY 2

1. INTRODUCTION 5

1.1 Database Design: 5
1.2 Database Application Development: 5
1.3 Optimization and Performance: 5

2. DATA REQUIREMENTS 5

2.1 Thesis Attributes 5
2.2 Relationships and Constraints 6

3. ER DIAGRAM IN CROW'S FOOT NOTATION 7

3.1 Entities 7
3.2 Relationships 8
3.3 Plan for Crow's Foot Notation 8
3.4 ER Diagram 9

4. RELATIONAL DATABASE DIAGRAM 9

4.1 Diagram Code Part 10

5. TABLE DEFINITIONS AND SQL SCRIPTS 12

5.1 University Table 12
5.2 Institute Table 12
5.3 Author Table 13
5.4 Supervisor Table 13
5.5 Subject Table 14
5.6 Keyword Table 14
5.7 Thesis Table 15
5.8 ThesisSupervisor Table 15
5.9 ThesisSubject Table 16
5.10 ThesisKeyword Table 16

6. SAMPLE DATA AND POPULATING COMMANDS 17

6.1 University Table 17
6.2 Institute Table 18
6.3 Author Table 19
6.4 Supervisor Table 20
6.5 Subject Table 21
6.6 Keyword Table 22
6.7 Thesis Table 24
6.8 ThesisSupervisor Table 25
6.9 ThesisSubject Table 26
6.10 ThesisKeyword Table 27

7. FIRST PART - CONCLUSION 29

7.2 Key Achievements 29
7.3 First Part - Final Remarks 29

8. DESIGN AND IMPLEMENTATION 31

8.1 View of Pages 31

8.1.1 Login Page *31*
8.1.2 Home Page *32*
8.1.3 Search Theses Page *32*
8.1.4 Manage Theses Page *33*
8.1.5 Add Entities Page *34*
8.1.6 Delete Entities Page *34*
8.2 Detailly Description of Pages 35
8.2.1 Login Page *35*
8.2.2 Home Page *36*
8.2.3 Search Theses Page *37*
8.2.4 Manage Theses Page *41*
8.2.5 Add New Entities Page *43*
8.2.6 Delete Entities Page *44*

9. DATABASE ACCESS AND MANIPULATION 47

9.1 Database Access 47
9.2 Common Operations 47
9.3 Challenges Encountered 48
9.4 Techniques for Data Manipulation 48
9.4.1 Transactions *48*
9.4.2 Query Optimization *49*
9.4.3 Object Relational Mapping (ORM) *49*
9.5 Advantages of Our Approach 49
9.5.1 Scalability: *49*
9.5.2 Data Integrity: *49*
9.5.3 Security: *50*
9.5.4 Ease of Use: *50*

10. APPENDIX 50

10.1 Source Codes 50
10.1.1 Front-end Code *50*
10.1.2 Back-end Code *71*
10.2 Self-Reflection 78

1. Introduction

The Graduate Thesis System (GTS) is designed to store, manage, and facilitate access to graduate theses efficiently. This system aims to streamline academic processes by organizing thesis data in a structured and accessible manner.

In this project, a database design and an associated database application will be developed for the GTS. The objectives of this project are as follows:

1.1 Database Design: Modeling entities and relationships in accordance with the specified business rules and requirements.

1.2 Database Application Development: Creating an application that enables users to easily add, update, and search for theses.

1.3 Optimization and Performance: Utilizing appropriate indexes and data structures to enhance database performance.

This document provides a comprehensive explanation of the methods and decisions made during the design phase. It also includes SQL scripts used to implement the database and sample data to demonstrate the system's functionality.

2. Data Requirements

The database for the Graduate Thesis System (GTS) must adhere to specific business rules and requirements to ensure proper functionality and data integrity. The following are the key data requirements:

2.1 Thesis Attributes

Each thesis must include the following mandatory attributes:

- **Thesis No:** A unique identifier for each thesis (numeric, whole number, maximum of 7 digits).
- **Title:** The title of the thesis (up to 500 characters).
- **Abstract:** A brief description of the thesis (up to 5000 characters).
- **Author:** The person who wrote the thesis.
- **Year:** The year the thesis was completed.
- **Type:** The type of thesis, which can be one of the following:
- Master
- Doctorate
- Specialization in Medicine

- Proficiency in Art
- **University:** The university where the thesis was completed.
- **Institute:** The institute associated with the university.
- **Number of Pages:** The total number of pages in the thesis.
- **Language:** The language in which the thesis was written (e.g., Turkish, English, French).
- **Submission Date:** The date the thesis was submitted.

2.2 Relationships and Constraints

1. **Author Relationship:**
- A person can author multiple theses (e.g., a Master's thesis followed by a Doctorate thesis).
- Each thesis can only have one author.

2. **Supervisor Relationship:**
- A thesis must have at least one supervisor.
- A thesis may have one optional co-supervisor.
- A person can supervise or co-supervise multiple theses.

3. **University-Institute Relationship:**
- A university can have multiple institutes.
- Each institute belongs to only one university.

4. **Subject Topics and Keywords:**
- A thesis must be associated with one or more predefined subject topics.
- Users select subject topics from a predefined list.
- A thesis can also have zero or more freely entered keywords for further classification.

5. **Language Constraint:**
- A thesis can be written in only one language, chosen from a predefined list (e.g., Turkish, English, French, etc.).

These data requirements form the foundation of the GTS database design. They ensure the system accurately captures and represents graduate thesis data while maintaining data integrity and consistency.

3. ER Diagram in Crow's Foot Notation

3.1 Entities

a) **Thesis**

- **Attributes**:
- ThesisID (Primary Key)
- Title
- Abstract
- AuthorID (Foreign Key)
- Year
- Type (Master, Doctorate, etc.)
- UniversityID (Foreign Key)
- InstituteID (Foreign Key)
- NumberOfPages
- Language
- SubmissionDate

b) **Author**

- **Attributes**:
- AuthorID (Primary Key)
- Name
- Surname

c) **Supervisor**

- **Attributes**:
- SupervisorID (Primary Key)
- Name
- Surname

d) **University**

- **Attributes**:
- UniversityID (Primary Key)
- Name

e) **Institute**

- **Attributes**:
- InstituteID (Primary Key)
- Name
- UniversityID (Foreign Key)

f) **Subject**

- **Attributes**:

- SubjectID (Primary Key)
- Name

g) **Keyword**

- **Attributes**:
- KeywordID (Primary Key)
- Word

3.2 Relationships

1. **Thesis - Author**:

- A thesis can belong to only one author (1:1).
- An author can write multiple theses (1:N).

2. **Thesis - Supervisor**:

- A thesis can have at least one and at most two supervisors (1:N).
- A supervisor can oversee multiple theses (N:N).
- *A junction table will be used for this relationship: ThesisSupervisor.*

3. **University - Institute**:

- A university can have multiple institutes (1:N).
- An institute is affiliated with only one university (N:1).

4. **Thesis - Subject**:

- A thesis can belong to multiple subjects (N:N).
- *A junction table will be used for this relationship: ThesisSubject.*

5. **Thesis - Keyword**:

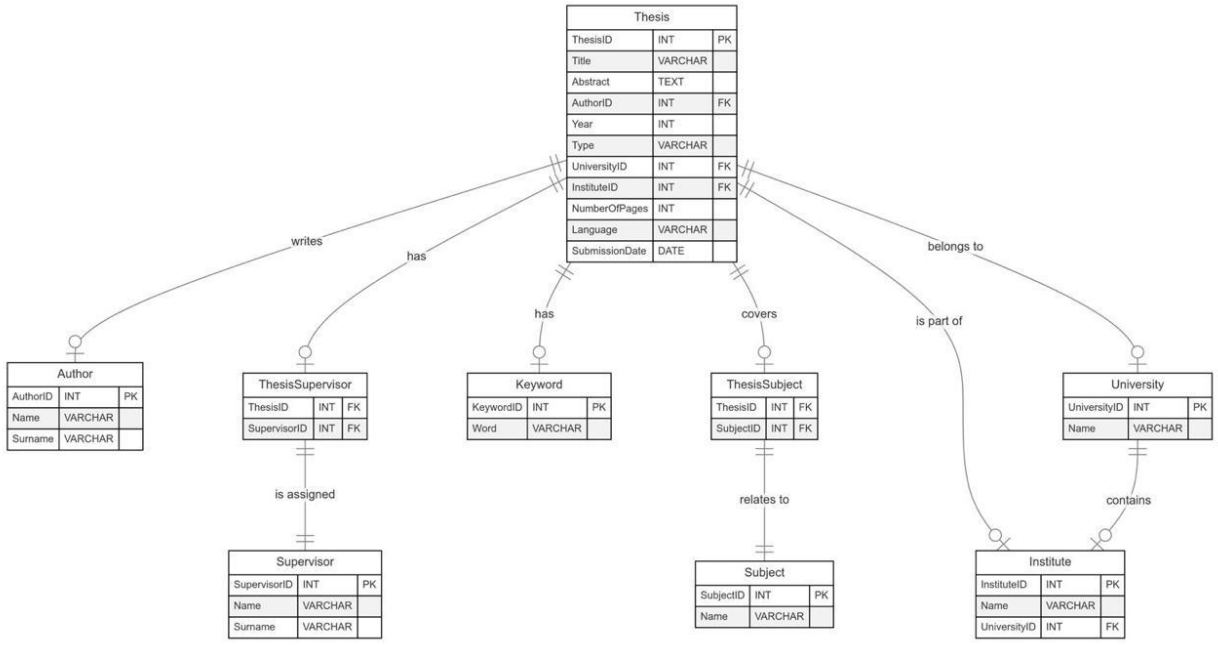
- A thesis can have zero or more keywords (1:N).

3.3 Plan for Crow's Foot Notation

To draw the ER diagram, the following connections will be established:

- **Thesis**:
- Thesis → Author (1:N)
- Thesis → Supervisor (N:N, junction table: ThesisSupervisor)
- Thesis → Subject (N:N, junction table: ThesisSubject)
- Thesis → Keyword (1:N)
- Thesis → University (N:1)
- Thesis → Institute (N:1)
- **University → Institute**: (1:N)

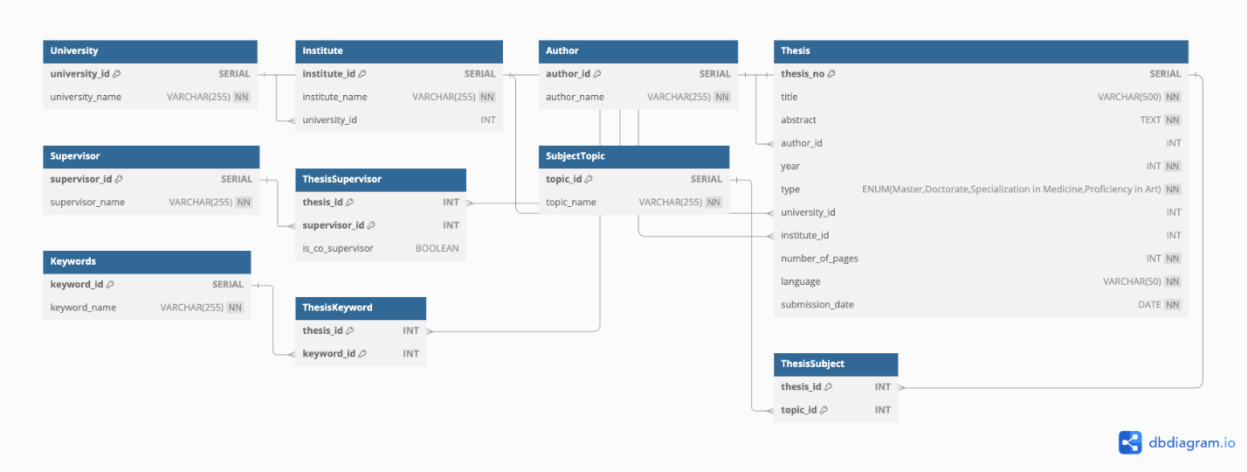
3.4 ER Diagram



4. Relational Database Diagram

The **Relational Database Diagram** represents the detailed mapping of entities and their relationships into relational database tables. This section includes a comprehensive schema to ensure all business rules and data requirements are captured effectively.

The diagram image shown below was created using dbdiagram.io and the provided code.



4.1 Diagram Code Part

```
Table University {
  university_id SERIAL [pk]
  university_name VARCHAR(255) [unique, not null]
}

Table Institute {
  institute_id SERIAL [pk]
  institute_name VARCHAR(255) [not null]
  university_id INT [ref: > University.university_id]
}

Table Author {
  author_id SERIAL [pk]
  author_name VARCHAR(255) [not null]
}

Table Thesis {
  thesis_no SERIAL [pk]
  title VARCHAR(500) [not null]
  abstract TEXT [not null]
  author_id INT [ref: > Author.author_id]
  year INT [not null]
  type ENUM('Master', 'Doctorate', 'Specialization in Medicine', 'Proficiency in Art') [not
null]
  university_id INT [ref: > University.university_id]
  institute_id INT [ref: > Institute.institute_id]
  number_of_pages INT [not null]
  language VARCHAR(50) [not null]
  submission_date DATE [not null]
}

Table Supervisor {
  supervisor_id SERIAL [pk]
  supervisor_name VARCHAR(255) [not null]
}

Table ThesisSupervisor {
  thesis_id INT [ref: > Thesis.thesis_no]
  supervisor_id INT [ref: > Supervisor.supervisor_id]
```

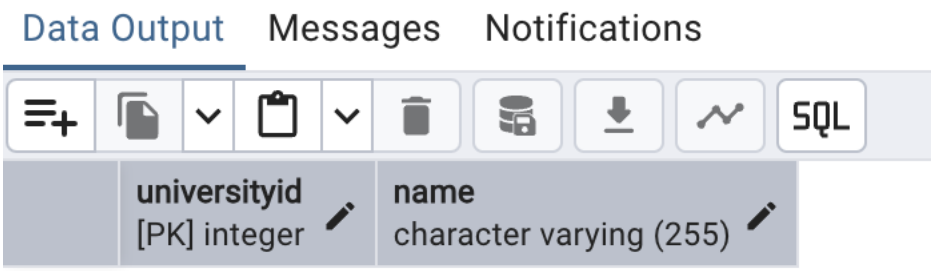
```
  is_co_supervisor BOOLEAN
  indexes {
    (thesis_id, supervisor_id) [pk]
  }
}

Table SubjectTopic {
  topic_id SERIAL [pk]
  topic_name VARCHAR(255) [unique, not null]
}

Table ThesisSubject {
  thesis_id INT [ref: > Thesis.thesis_no]
  topic_id INT [ref: > SubjectTopic.topic_id]
  indexes {
    (thesis_id, topic_id) [pk]
  }
}

Table Keywords {
  keyword_id SERIAL [pk]
  keyword_name VARCHAR(255) [not null]
}

Table ThesisKeyword {
  thesis_id INT [ref: > Thesis.thesis_no]
  keyword_id INT [ref: > Keywords.keyword_id]
  indexes {
    (thesis_id, keyword_id) [pk]
  }
}
```

5. Table Definitions and SQL Scripts

Create commands you have used to create entities, relationships.

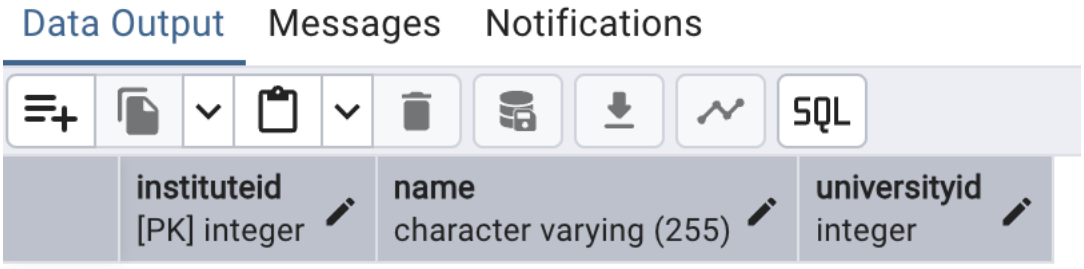
5.1 University Table

```
CREATE TABLE University (
  UniversityID SERIAL PRIMARY KEY,
  Name VARCHAR(255) NOT NULL
);
```



5.2 Institute Table

```
CREATE TABLE Institute (
  InstituteID SERIAL PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  UniversityID INT NOT NULL,
  FOREIGN KEY (UniversityID) REFERENCES University(UniversityID)
);
```



5.3 Author Table

```
CREATE TABLE Author (
  AuthorID SERIAL PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Surname VARCHAR(100) NOT NULL
);
```

Data Output | Messages | Notifications

	authorid [PK] integer	name character varying (100)	surname character varying (100)

5.4 Supervisor Table

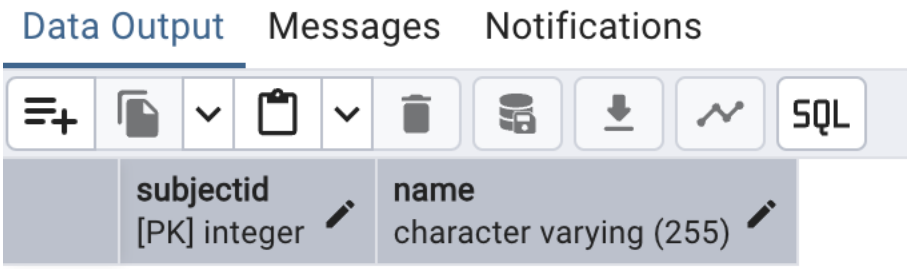
```
CREATE TABLE Supervisor (
  SupervisorID SERIAL PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Surname VARCHAR(100) NOT NULL
);
```

Data Output | Messages | Notifications

	supervisorid [PK] integer	name character varying (100)	surname character varying (100)

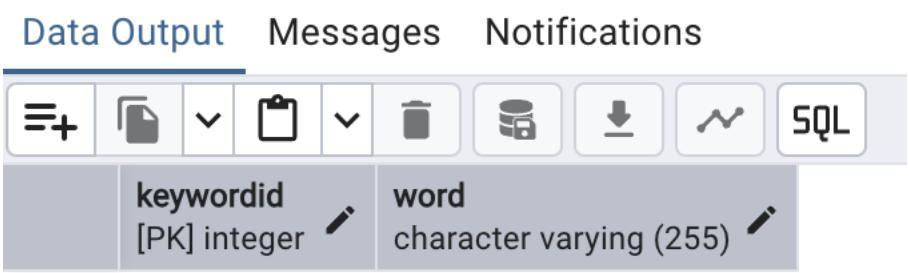
5.5 Subject Table

```
CREATE TABLE Subject (
  SubjectID SERIAL PRIMARY KEY,
  Name VARCHAR(255) NOT NULL
);
```



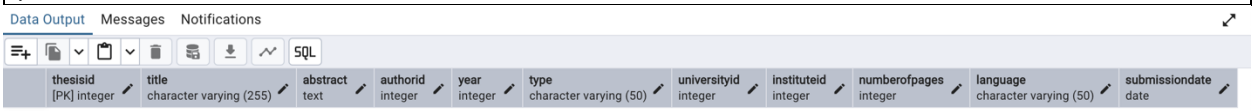
5.6 Keyword Table

```
CREATE TABLE Keyword (
  KeywordID SERIAL PRIMARY KEY,
  Word VARCHAR(255) NOT NULL
);
```



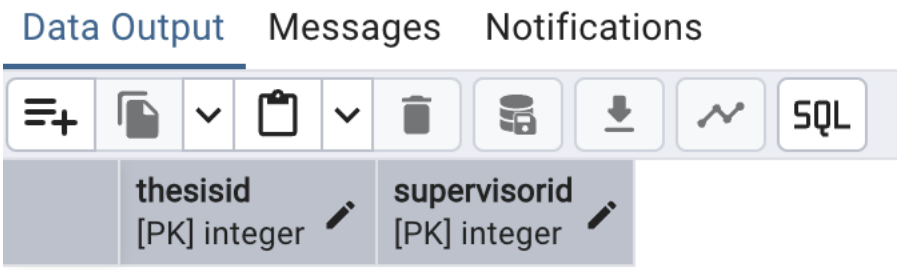
5.7 Thesis Table

```
CREATE TABLE Thesis (
  ThesisID SERIAL PRIMARY KEY,
  Title VARCHAR(255) NOT NULL,
  Abstract TEXT,
  AuthorID INT NOT NULL,
  Year INT NOT NULL,
  Type VARCHAR(50),
  UniversityID INT NOT NULL,
  InstituteID INT NOT NULL,
  NumberOfPages INT,
  Language VARCHAR(50),
  SubmissionDate DATE NOT NULL,
  FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),
  FOREIGN KEY (UniversityID) REFERENCES University(UniversityID),
  FOREIGN KEY (InstituteID) REFERENCES Institute(InstituteID)
);
```



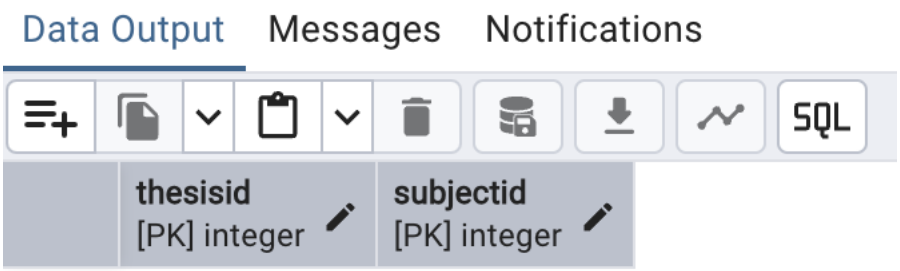
5.8 ThesisSupervisor Table

```
CREATE TABLE ThesisSupervisor (
  ThesisID INT NOT NULL,
  SupervisorID INT NOT NULL,
  PRIMARY KEY (ThesisID, SupervisorID),
  FOREIGN KEY (ThesisID) REFERENCES Thesis(ThesisID),
  FOREIGN KEY (SupervisorID) REFERENCES Supervisor(SupervisorID)
);
```



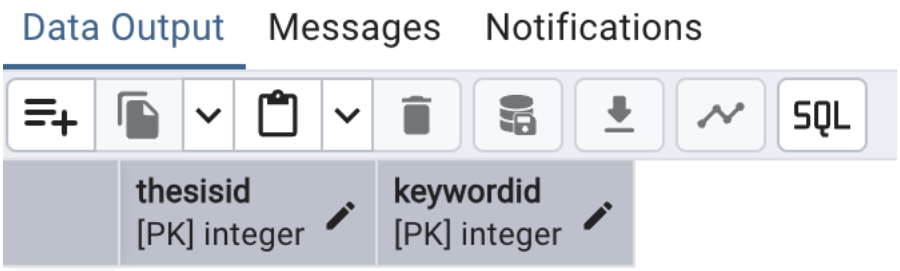
5.9 ThesisSubject Table

```
CREATE TABLE ThesisSubject (
  ThesisID INT NOT NULL,
  SubjectID INT NOT NULL,
  PRIMARY KEY (ThesisID, SubjectID),
  FOREIGN KEY (ThesisID) REFERENCES Thesis(ThesisID),
  FOREIGN KEY (SubjectID) REFERENCES Subject(SubjectID)
);
```



5.10 ThesisKeyword Table

```
CREATE TABLE ThesisKeyword (
  ThesisID INT NOT NULL,
  KeywordID INT NOT NULL,
  PRIMARY KEY (ThesisID, KeywordID),
  FOREIGN KEY (ThesisID) REFERENCES Thesis(ThesisID),
  FOREIGN KEY (KeywordID) REFERENCES Keyword(KeywordID)
);
```



6. Sample Data and Populating Commands

Insert Into commands you have used to populate the database.

6.1 University Table

```
INSERT INTO University (Name) VALUES
('Boğaziçi Üniversitesi'),
('İstanbul Teknik Üniversitesi'),
('Hacettepe Üniversitesi'),
('Orta Doğu Teknik Üniversitesi'),
('Marmara Üniversitesi'),
('Ankara Üniversitesi'),
('Ege Üniversitesi');
```

Data Output | Messages | Notifications

	universityid [PK] integer	name character varying (255)
1	57	Boğaziçi Üniversitesi
2	58	İstanbul Teknik Üniversitesi
3	59	Hacettepe Üniversitesi
4	60	Orta Doğu Teknik Üniversitesi
5	61	Marmara Üniversitesi
6	62	Ankara Üniversitesi
7	63	Ege Üniversitesi

6.2 Institute Table

```
INSERT INTO Institute (Name, UniversityID) VALUES
(1, 'Sosyal Bilimler Enstitüsü', 57),
(2, 'Fen Bilimleri Enstitüsü', 58),
(3, 'Eğitim Bilimleri Enstitüsü', 59),
(4, 'Sağlık Bilimleri Enstitüsü', 60),
(5, 'Uygulamalı Bilimler Enstitüsü', 61),
(6, 'Adalet Meslek Enstitüsü', 62),
(7, 'Yazılım ve Bilişim Enstitüsü', 63);
```

Data Output | Messages | Notifications

	instituteid [PK] integer	name character varying (255)	universityid integer
1	1	Sosyal Bilimler Enstitüsü	57
2	2	Fen Bilimleri Enstitüsü	58
3	3	Eğitim Bilimleri Enstitüsü	59
4	4	Sağlık Bilimleri Enstitüsü	60
5	5	Uygulamalı Bilimler Enstitüsü	61
6	7	Adalet Meslek Enstitüsü	62
7	8	Yazılım ve Bilişim Enstitüsü	63

6.3 Author Table

```
INSERT INTO Author (Name, Surname) VALUES
('Ahmet', 'Yılmaz'),
('Ayşe', 'Kaya'),
('Mehmet', 'Demir'),
('Fatma', 'Çelik'),
('Ali', 'Şahin'),
('Zeynep', 'Aksoy'),
('Emre', 'Güneş');
```

Data Output | Messages | Notifications

	authorid [PK] integer	name character varying (100)	surname character varying (100)
1	8	Ahmet	Yılmaz
2	9	Ayşe	Kaya
3	10	Mehmet	Demir
4	11	Fatma	Çelik
5	12	Ali	Şahin
6	13	Zeynep	Aksoy
7	14	Emre	Güneş

6.4 Supervisor Table

```
INSERT INTO Supervisor (Name, Surname) VALUES
('Haluk', 'Özdemir'),
('Elif', 'Arslan'),
('Kerem', 'Taş'),
('Leyla', 'Aydın'),
('Cengiz', 'Yalçın'),
('Aylin', 'Koç'),
('Burak', 'Karaca');
```

Data Output | Messages | Notifications

	supervisorid [PK] integer	name character varying (100)	surname character varying (100)
1	8	Haluk	Özdemir
2	9	Elif	Arslan
3	10	Kerem	Taş
4	11	Leyla	Aydın
5	12	Cengiz	Yalçın
6	13	Aylin	Koç
7	14	Burak	Karaca

6.5 Subject Table

```
INSERT INTO Subject (Name) VALUES
('Makine Öğrenimi'),
('Veri Madenciliği'),
('Yapay Zeka'),
('Genetik Algoritmalar'),
('Kuantum Hesaplama'),
('Robotik'),
('Nörobilim');
```

Data Output | Messages | Notifications

	subjectid [PK] integer	name character varying (255)
1	8	Makine Öğrenimi
2	9	Veri Madenciliği
3	10	Yapay Zeka
4	11	Genetik Algoritmalar
5	12	Kuantum Hesaplama
6	13	Robotik
7	14	Nörobilim

6.6 Keyword Table

```
INSERT INTO Keyword (Word) VALUES
('Algorithms'),
('Data Analytics'),
('Deep Learning'),
('Big Data'),
('Blockchain'),
('Optimization'),
('Signal Processing'),
('Artificial Intelligence'),
('Machine Learning'),
('Quantum Technology'),
('Robotic Systems'),
('Medical Imaging'),
('Natural Language Processing'),
('Autonomous Vehicles'),
('Bioinformatics'),
('Database Management'),
('Cloud Computing'),
('Cybersecurity'),
('Educational Technologies'),
('Energy Systems'),
('Cryptography');
```

Data Output | Messages | Notifications

	keywordid [PK] integer	word character varying (255)
1	22	Algorithms
2	23	Data Analytics
3	24	Deep Learning
4	25	Big Data
5	26	Blockchain
6	27	Optimization
7	28	Signal Processing
8	29	Artificial Intelligence
9	30	Machine Learning
10	31	Quantum Technology
11	32	Robotic Systems
12	33	Medical Imaging
13	34	Natural Language Processing
14	35	Autonomous Vehicles
15	36	Bioinformatics
16	37	Database Management
17	38	Cloud Computing
18	39	Cybersecurity
19	40	Educational Technologies
20	41	Energy Systems
21	42	Cryptography

6.7 Thesis Table

```
INSERT INTO Thesis (Title, Abstract, AuthorID, Year, Type, UniversityID, InstituteID,
NumberOfPages, Language, SubmissionDate) VALUES
('Yapay Zeka ile Hastalık Teşhisine Yönelik Araştırma',
'Bu çalışma, sağlık sektöründe yapay zeka algoritmalarının kullanımını araştırmaktadır.
Özellikle, sinir ağları ve derin öğrenme teknikleri kullanılarak kanser, diyabet gibi
hastalıkların erken teşhisine yönelik çözümler sunulmuştur. Çalışmada gerçek hasta
verileri kullanılarak kapsamlı testler yapılmış ve elde edilen sonuçlar istatistiksel olarak
analiz edilmiştir.',
8, 2024, 'Doctorate', 57, 1, 200, 'Türkçe', '2024-12-01'),

('Machine Learning for Image Processing',
'This thesis aims to integrate machine learning algorithms with image processing
techniques. Through this integration, innovative solutions for classification and
segmentation of image data are presented.',
9, 2023, 'Master', 58, 2, 150, 'English', '2023-11-15'),

('Blockchain et Sécurité',
'La technologie blockchain est devenue un outil clé pour résoudre les problèmes de
sécurité dans les systèmes décentralisés. Cette thèse explore l'architecture de sécurité
des systèmes blockchain.',
10, 2024, 'Master', 59, 3, 180, 'Français', '2024-10-05'),

('Técnicas de Optimización',
'Esta tesis examina técnicas de optimización utilizadas en contextos industriales y
académicos. Se analizan métodos como algoritmos genéticos, programación lineal y
programación dinámica.',
11, 2022, 'Doctorate', 60, 4, 220, 'Español', '2022-09-12'),

('Robotik ve Kontrol Sistemleri',
'Robotik alanındaki kontrol sistemlerinin gelişimini inceleyen bu tez, otonom sistemlerde
kullanılan kontrol algoritmalarını derinlemesine analiz etmektedir.',
12, 2023, 'Specialization in Medicine', 61, 5, 190, 'Türkçe', '2023-08-20'),

('Deep Learning in Neuroscience',
'This thesis investigates the application of deep learning algorithms in the field of
neuroscience.',
13, 2022, 'Doctorate', 62, 3, 210, 'English', '2022-07-15'),

('Fundamentos de la Computación Cuántica',
'Esta tesis analiza los principios fundamentales de la computación cuántica y cómo se
diferencian de los ordenadores clásicos.',
14, 2024, 'Master', 63, 7, 160, 'Español', '2024-05-10');
```

Data Output Messages Notifications

Showing rows: 1 to 7 Page No: 1 of 1

	thesisid [PK] inte	title character varying (255)	abstract text	authorid integer	year integer	type character varying (50)	universityid integer	instituteid integer	numberofpages integer	language character varying (50)	submissiondate date
1	43	Yapay Zeka ile Hastalık Teşhisi	Bu çalışma, sağlık sektöründe yapay...	8	2024	Doctorate	57	1	200	Türkçe	2024-12-01
2	44	Machine Learning for Image Processing	This thesis aims to integrate machin...	9	2023	Master	58	2	150	English	2023-11-15
3	45	Blockchain et Sécurité	La technologie blockchain est deven...	10	2024	Master	59	3	180	Français	2024-10-05
4	46	Técnicas de Optimización	Esta tesis examina técnicas de opti...	11	2022	Doctorate	60	4	220	Español	2022-09-12
5	47	Robotik ve Kontrol Sistemleri	Robotik alanındaki kontrol sistemleri...	12	2023	Specialization in M...	61	5	190	Türkçe	2023-08-20
6	48	Deep Learning in Neuroscience	This thesis investigates the applicati...	13	2022	Doctorate	62	3	210	English	2022-07-15
7	49	Fundamentos de la Computación Cuán...	Esta tesis analiza los principios fund...	14	2024	Master	63	7	160	Español	2024-05-10

6.8 ThesisSupervisor Table

```
INSERT INTO ThesisSupervisor (ThesisID, SupervisorID) VALUES
(43, 8),
(43, 9),
(44, 10),
(45, 11),
(45, 12),
(47, 13),
(48, 14);
```

Data Output Messages Notifications

	thesisid [PK] integer	supervisorid [PK] integer
1	43	8
2	43	9
3	44	10
4	45	11
5	45	12
6	47	13
7	48	14

6.9 ThesisSubject Table

```
INSERT INTO ThesisSubject (ThesisID, SubjectID) VALUES
(43, 8),
(44, 9),
(45, 10),
(46, 11),
(47, 12),
(48, 13),
(49, 14);
```

Data Output | Messages | Notifications

	thesisid [PK] integer	subjectid [PK] integer
1	43	8
2	44	9
3	45	10
4	46	11
5	47	12
6	48	13
7	49	14

6.10 ThesisKeyword Table

```
INSERT INTO ThesisKeyword (ThesisID, KeywordID) VALUES
(43, 22),
(43, 28),
(43, 29),
(44, 23),
(44, 24),
(44, 30),
(45, 26),
(45, 39),
(45, 42),
(46, 27),
(46, 36),
(46, 37),
(47, 32),
(47, 33),
(47, 30),
(48, 24),
(48, 28),
(48, 40),
(49, 31),
(49, 25),
(49, 42);
```

Data Output | Messages | Notifications

	thesisid [PK] integer	keywordid [PK] integer
1	43	22
2	43	28
3	43	29
4	44	23
5	44	24
6	44	30
7	45	26
8	45	39
9	45	42
10	46	27
11	46	36
12	46	37
13	47	32
14	47	33
15	47	30
16	48	24
17	48	28
18	48	40
19	49	31
20	49	25
21	49	42

7. First Part - Conclusion

The Graduate Thesis System (GTS) database design presented in this document adheres to the specified business rules and data requirements. The design ensures that all critical relationships and constraints are accurately represented to maintain data integrity and consistency.

7.2 Key Achievements

1. **Comprehensive Database Design:**
   - The database schema is structured to capture all necessary attributes and relationships, supporting the storage and retrieval of detailed graduate thesis data.
2. **Optimization for Performance:**
   - Proper indexing strategies were employed to enhance query performance for common operations like searching by title, author, or subject.
3. **Scalability and Extensibility:**
   - The relational schema is designed to accommodate additional data or requirements, such as new thesis types, languages, or institutes, without significant modifications.
4. **Sample Data for Testing:**
   - Meaningful sample data were populated in each table, demonstrating the database's functionality and preparing it for integration with the application.

7.3 First Part - Final Remarks

The GTS database provides a robust foundation for managing graduate thesis information. It ensures ease of use for both data entry and retrieval while maintaining strict adherence to the business rules. By building on this solid database design, the GTS application can deliver a comprehensive and user-friendly experience for managing graduate theses effectively.

maltepe university
istanbul www.maltepe.edu.tr

SE 307 – Database Management Systems

Term Project
Second Part – 2024/25 Fall

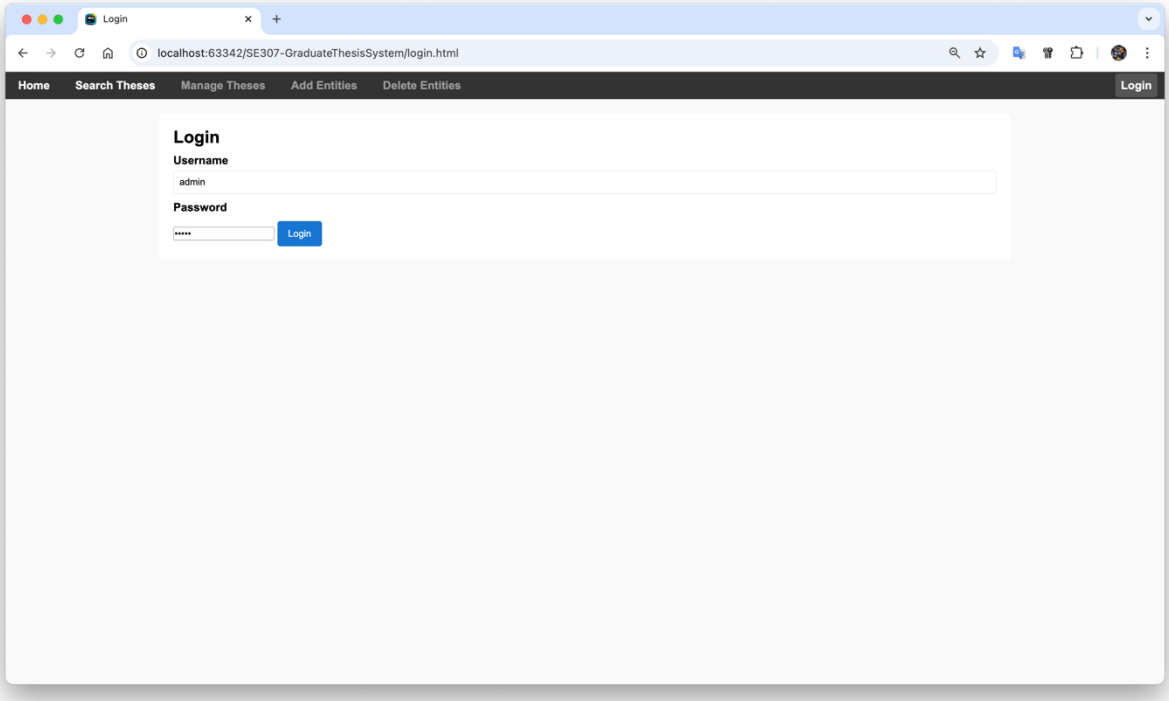
Assoc. Prof. Dr. Volkan TUNALI

Prepared by:
21 07 06 017 - Muhammed Numan Karaca
21 07 06 028 - Ömer Faruk Özer
21 07 06 039 - Berke Ensel

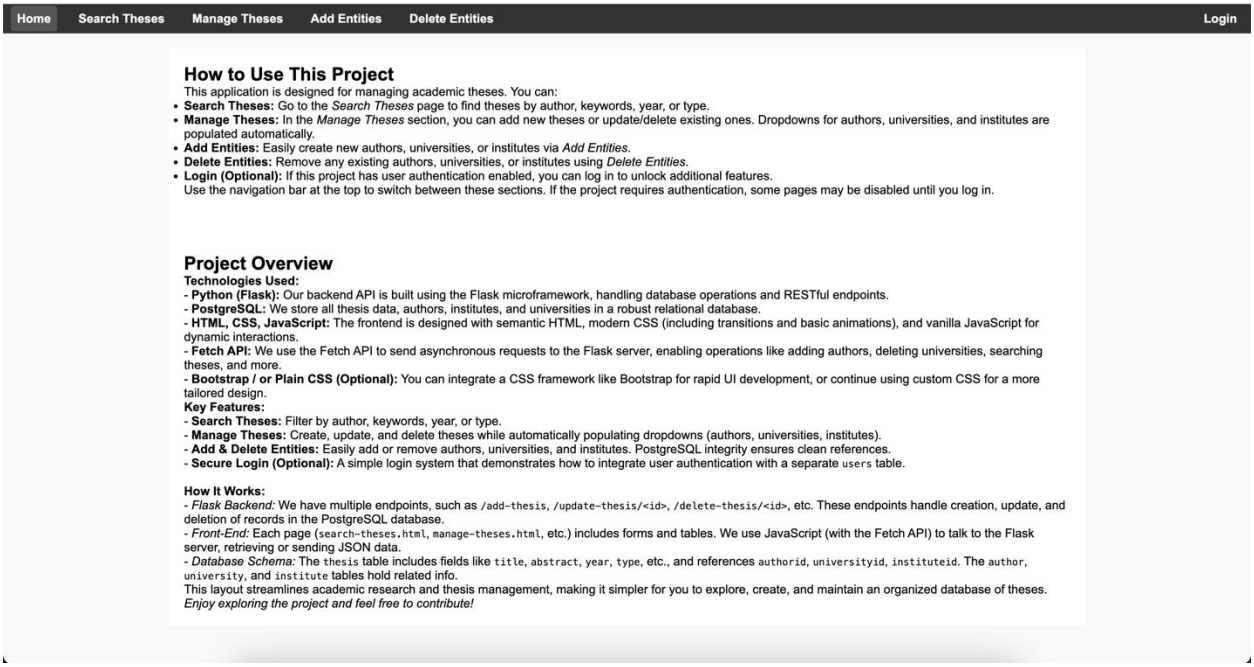
8. Design and Implementation

8.1 View of Pages

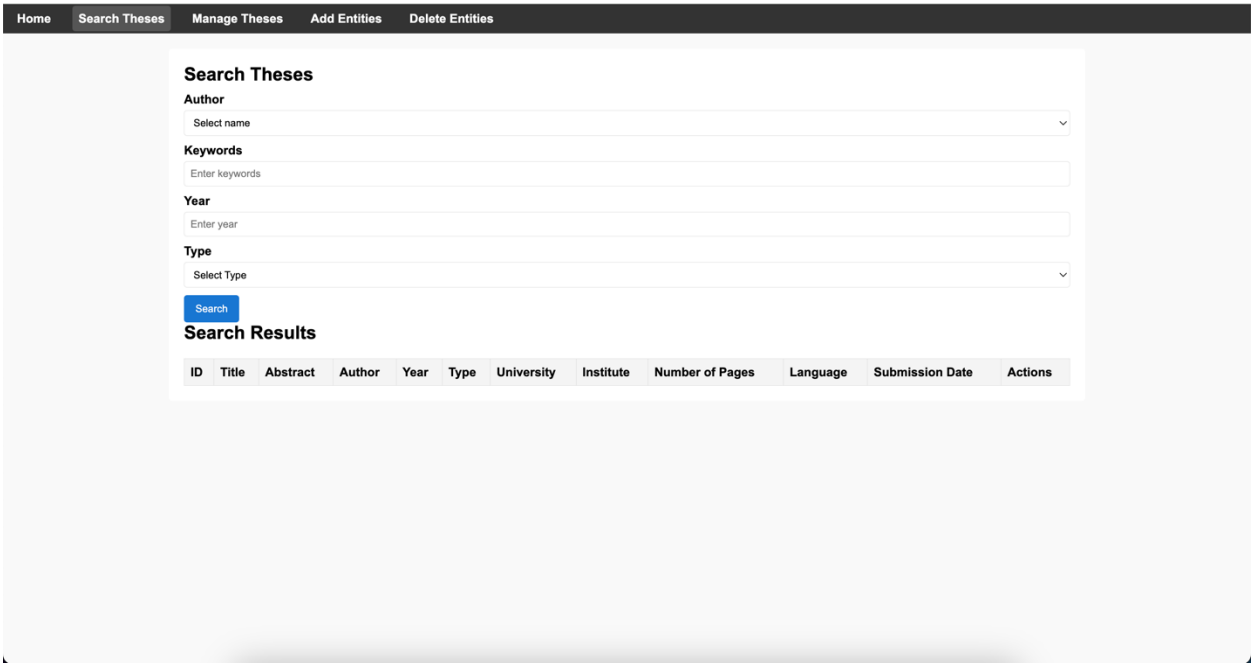
8.1.1 Login Page



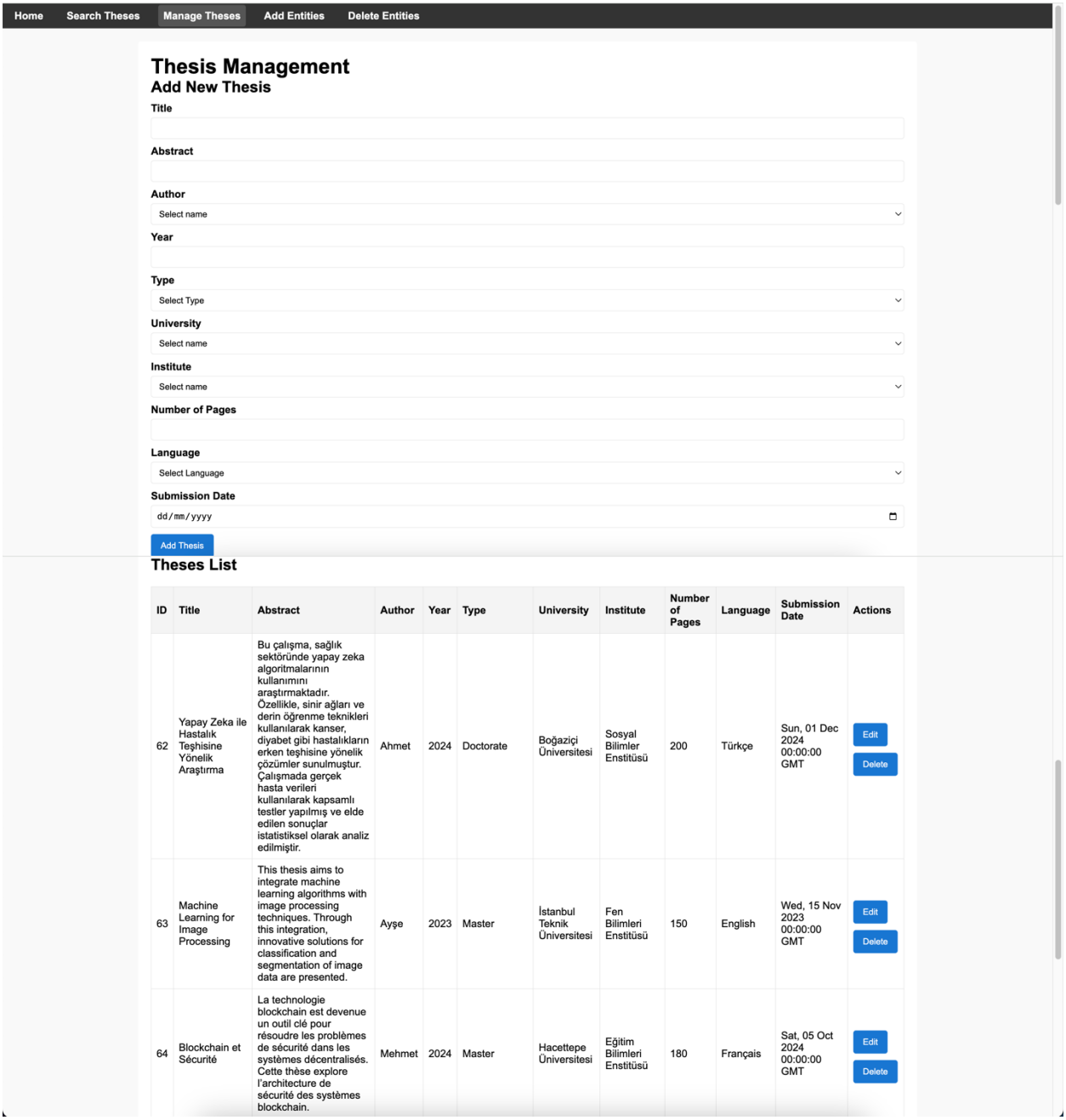
8.1.2 Home Page



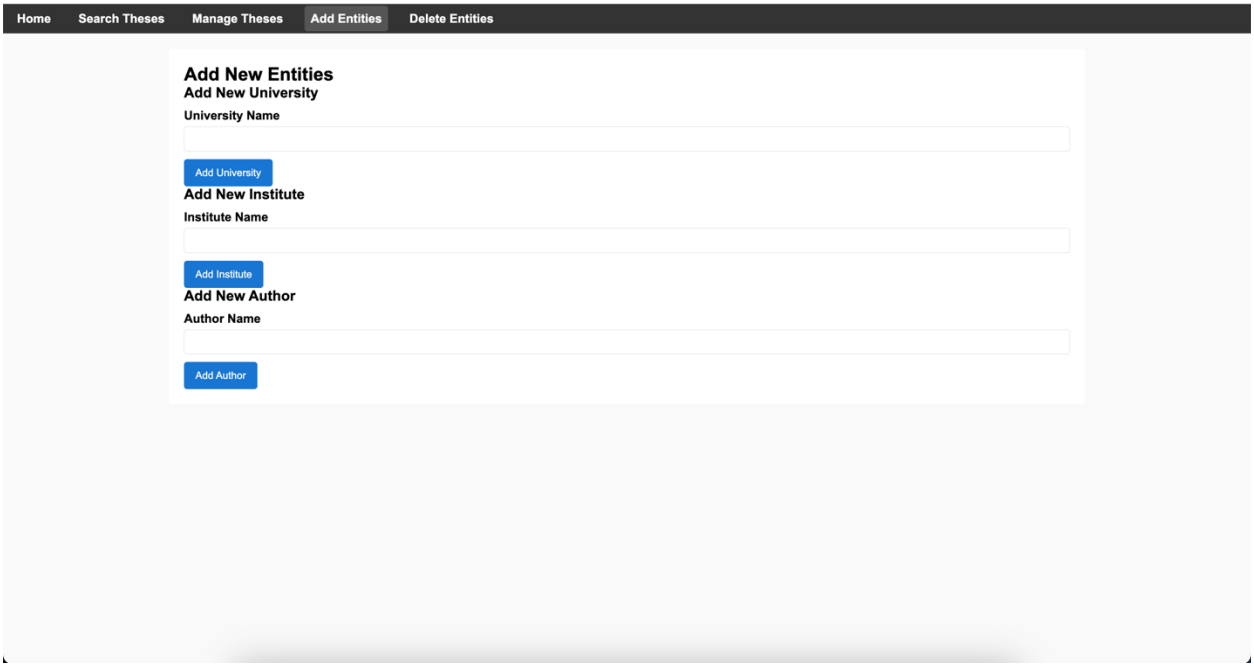
8.1.3 Search Theses Page



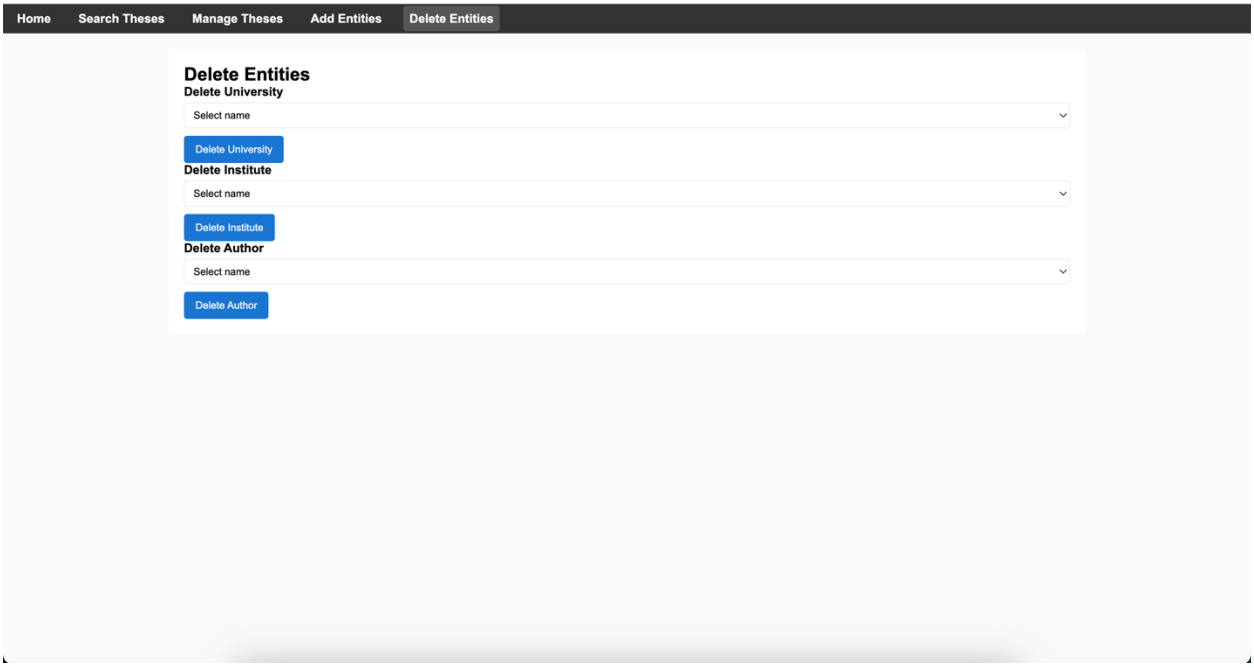
8.1.4 Manage Theses Page



8.1.5 Add Entities Page

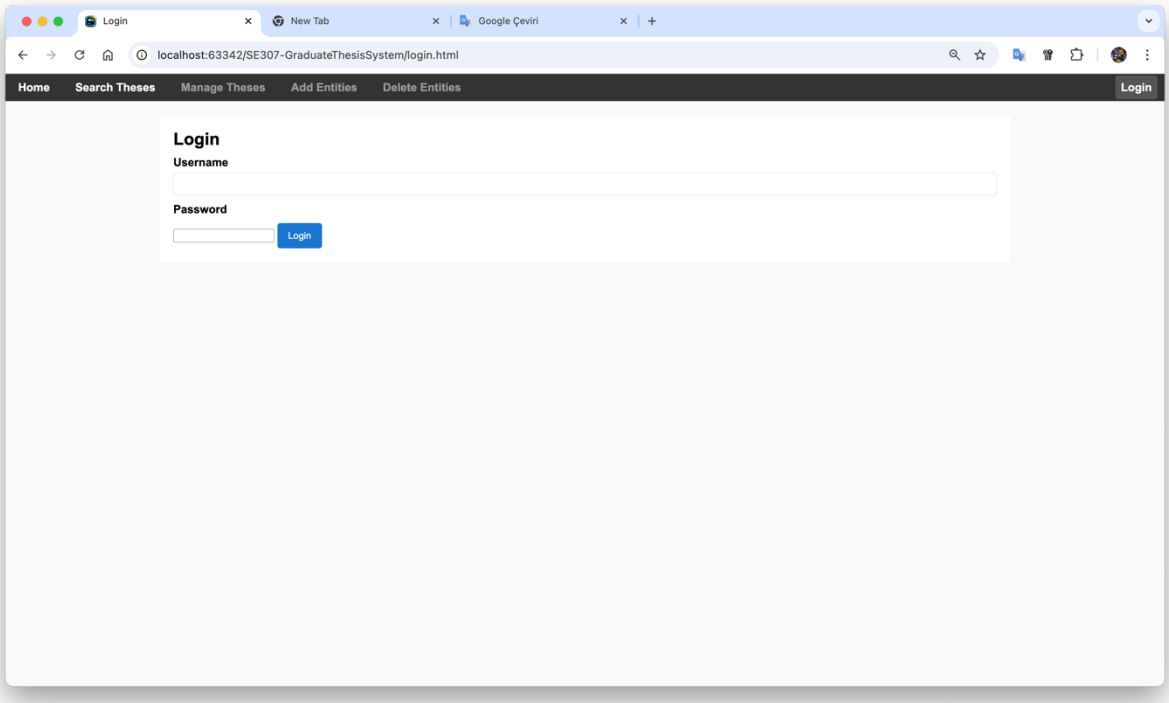


8.1.6 Delete Entities Page



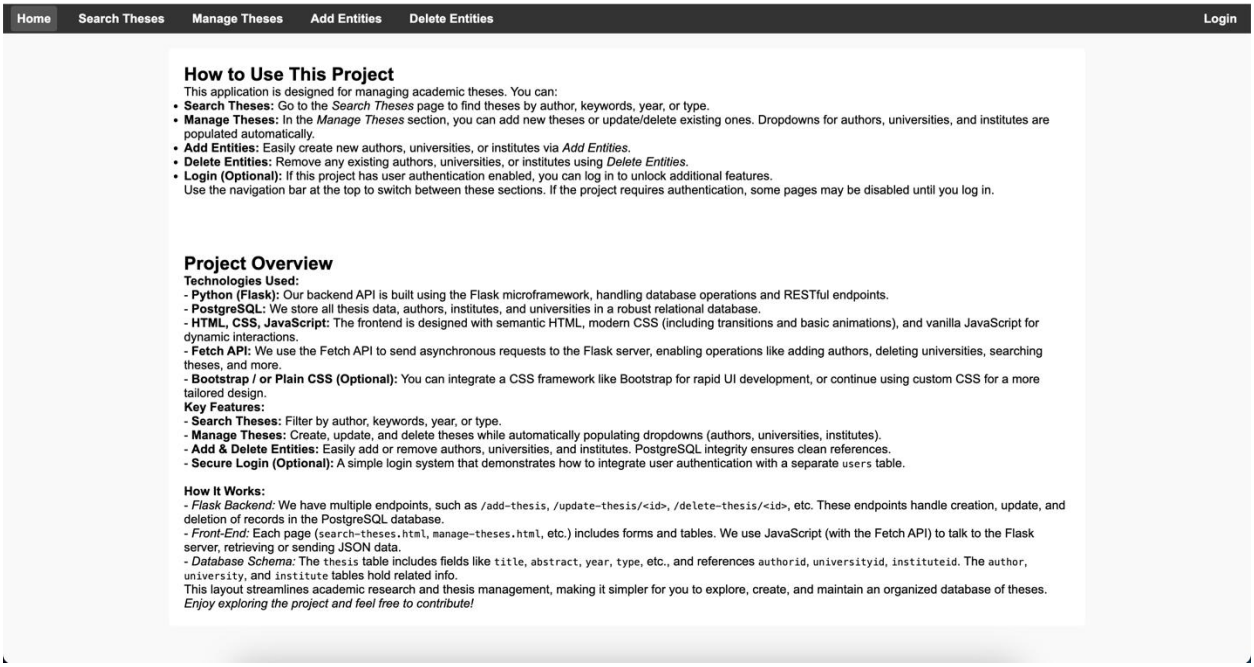
8.2 Detailly Description of Pages

8.2.1 Login Page



- By Logging in, you gain access to “Manage Theses.”

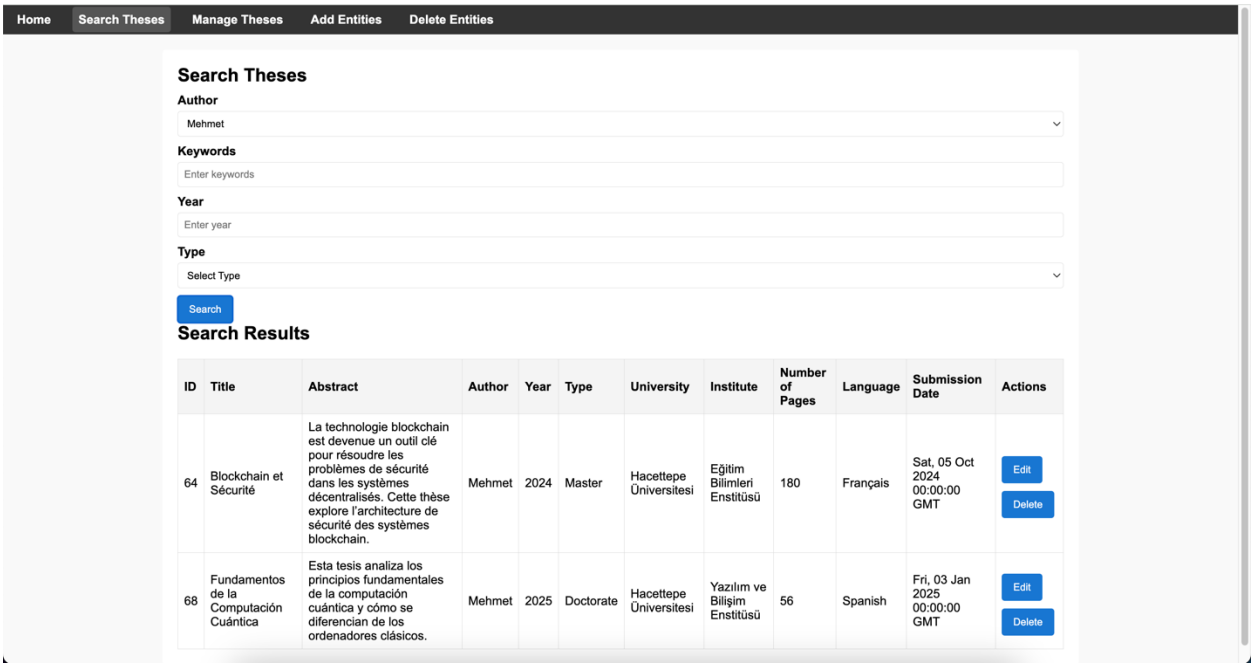
8.2.2 Home Page



- The first section displays how the project and the webpage function.
- The second section contains explanations of the technologies and techniques used in the project.

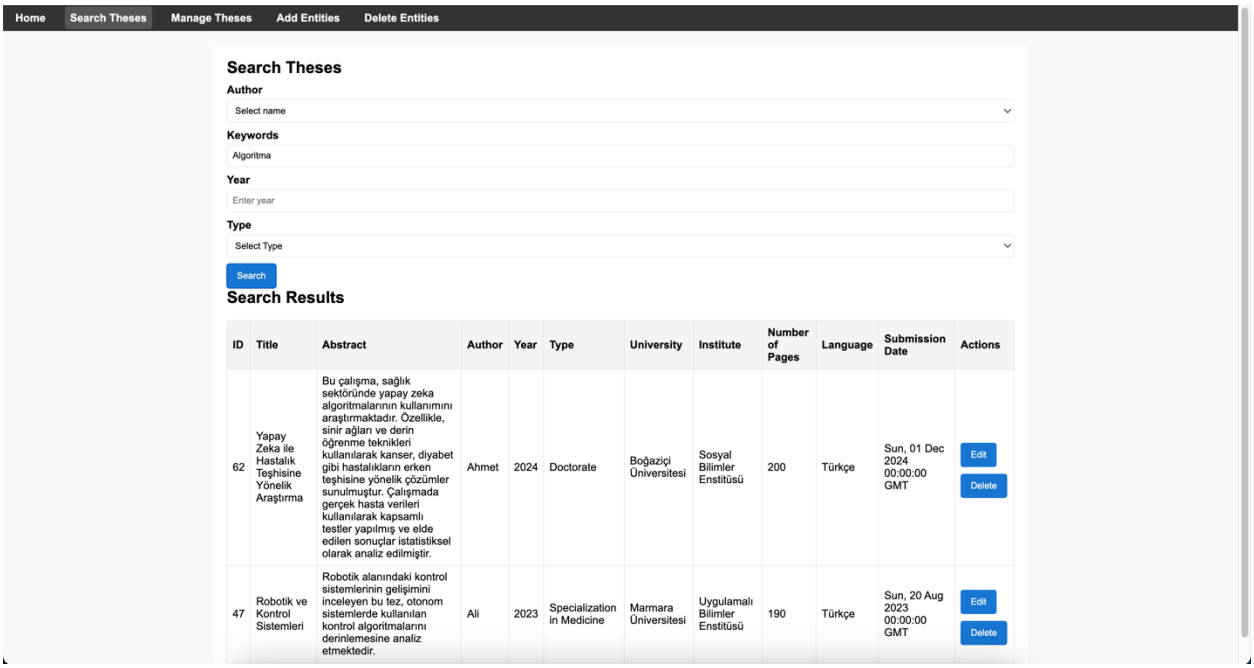
8.2.3 Search Theses Page

8.2.3.1 Author Searching



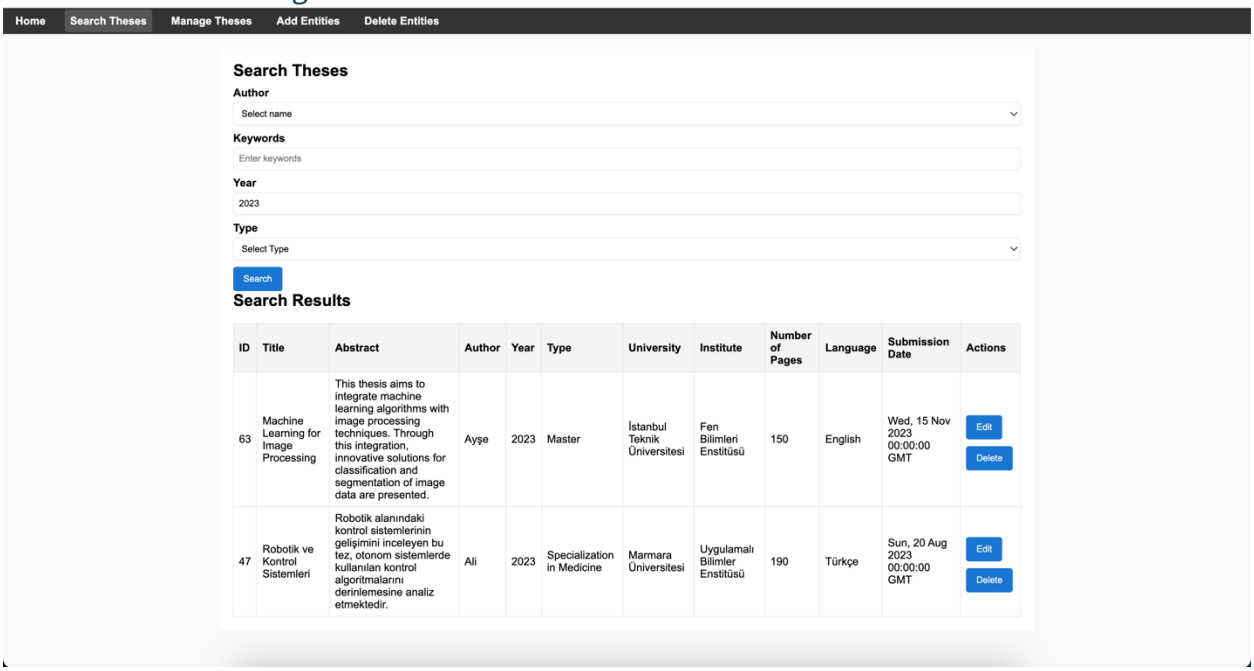
- On the “Search Theses” page, searching for an Author displays the Theses associated with that person.
- When we entered “Mehmet” in the Author field, the theses belonging to Mehmet were displayed.

8.2.3.2 Keywords Searching



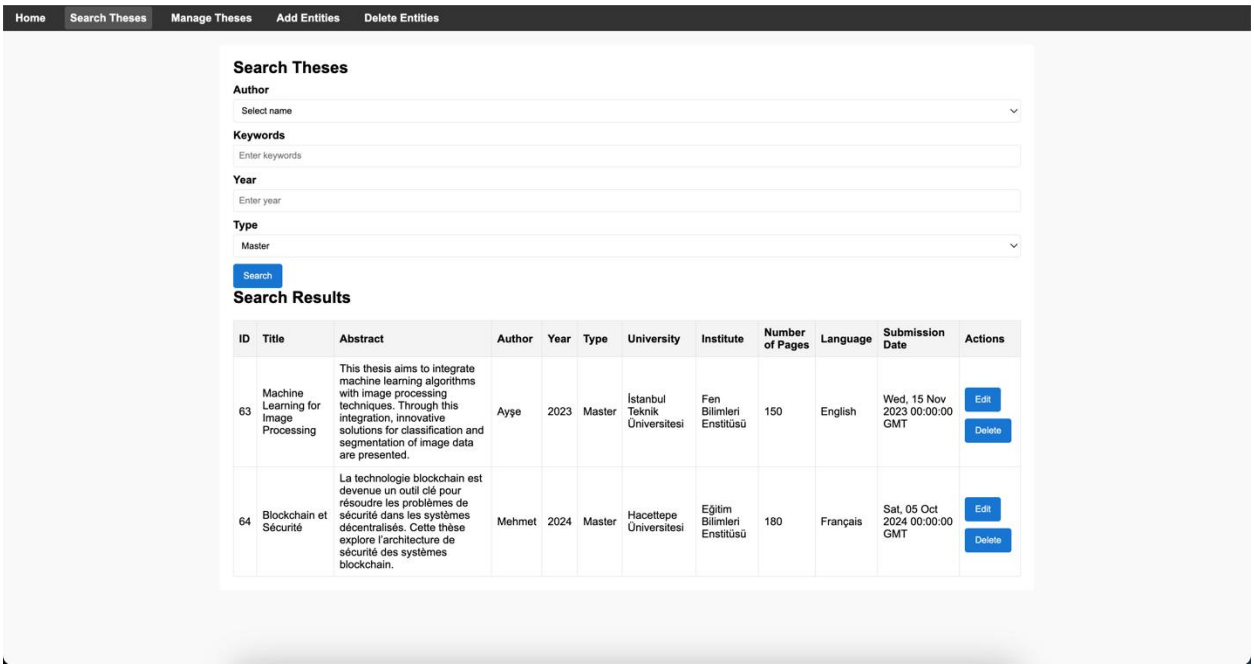
- On the “Search Theses” page, searching for an Keyword displays the Theses associated with that specific keyword.
- When we entered “Algoritma” in the Keyword field, the theses belonging to Keyword were displayed.

8.2.3.3 Year Searching



- On the “Search Theses” page, searching for an Year displays the Theses associated with that year.
- When we entered “2023” in the year field, the theses belonging to 2023 were displayed.

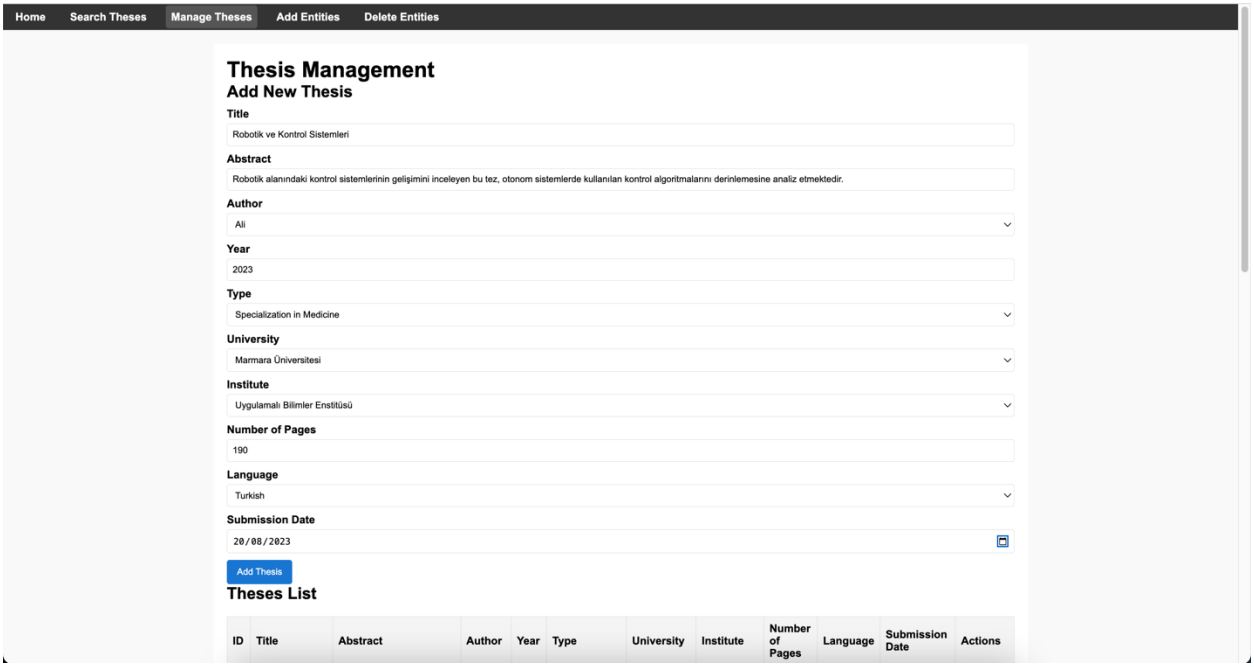
8.2.3.4 Type Searching



- On the “Search Theses” page, searching for an Type displays the Theses associated with that type.
- When we entered “Master” in the Type field, the theses belonging to Master were displayed.

8.2.4 Manage Theses Page

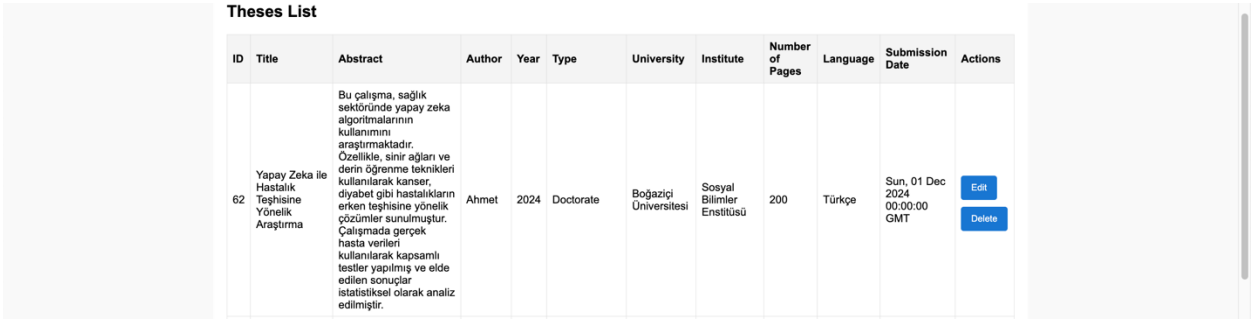
8.2.4.1 Adding New Thesis



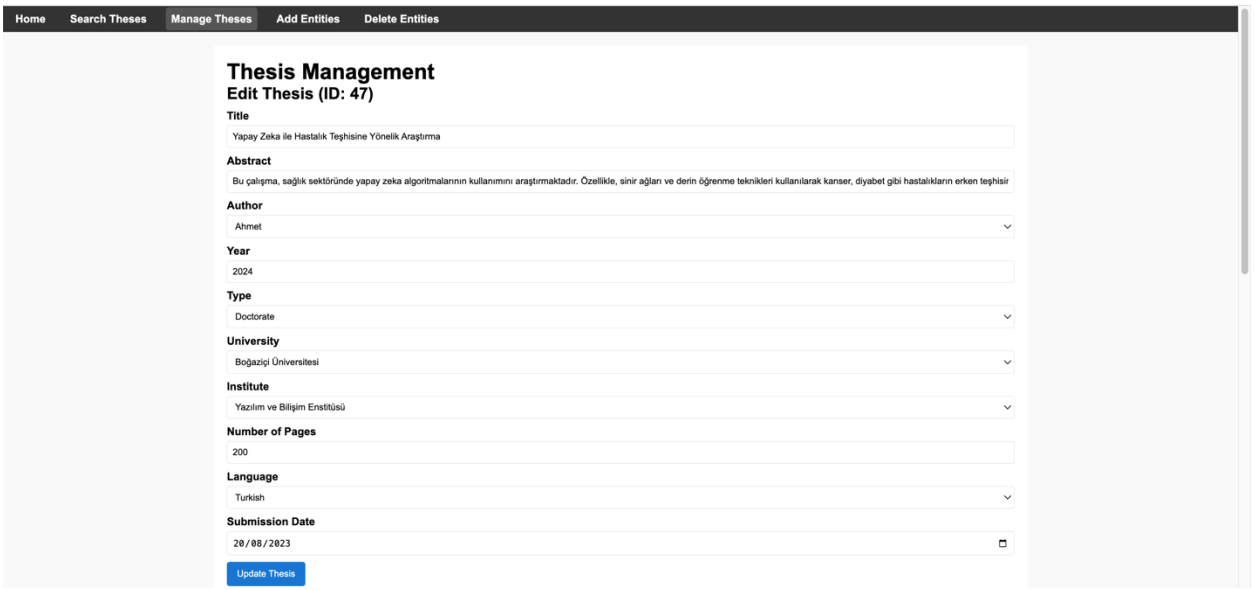
- First, we fill in the required information for the thesis and then click the “Add Thesis” button.

- Secondly, the newly added Thesis appears in the “Theses List” section.

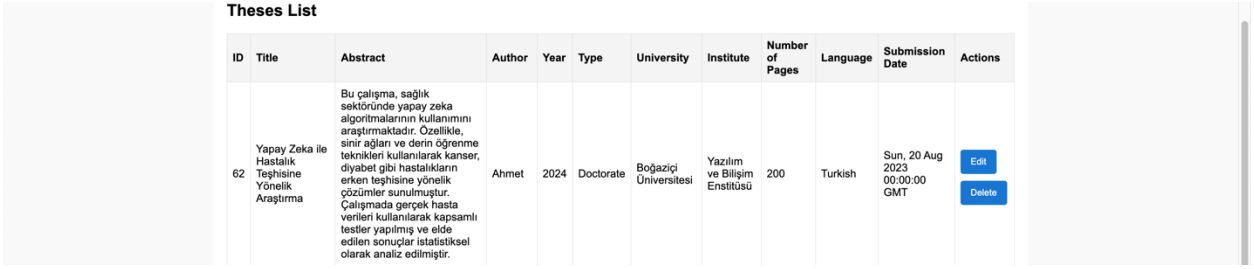
8.2.4.2 Editing Thesis



- Firstly, this is the old version of the thesis with ID 62.

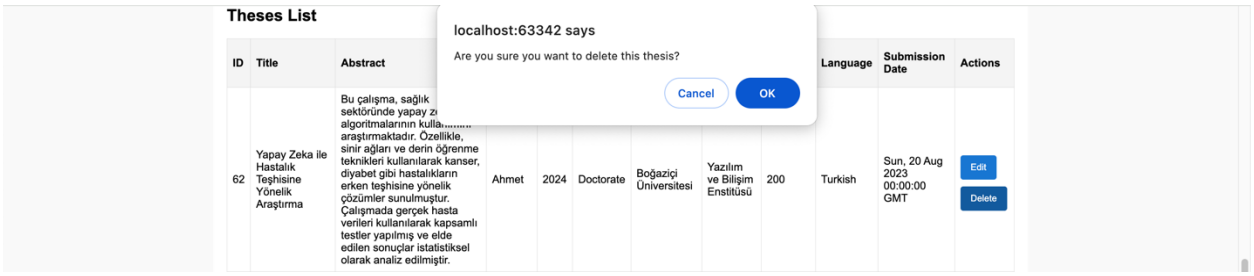


- Secondly, we update the parts of the thesis we want to modify on this screen.
- In this example, the fields we modified are the “Date” and the “Institute”.



- Lastly, this is the final version of the thesis with ID 62.

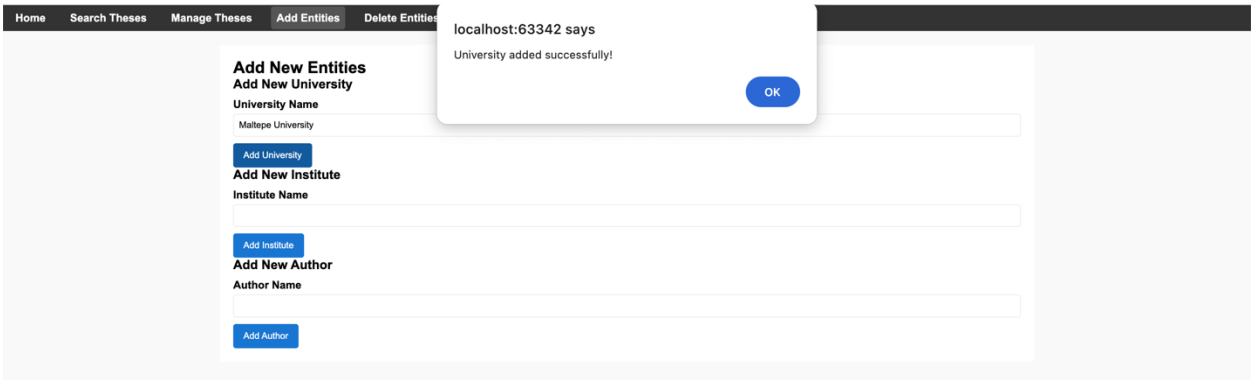
8.2.4.3 Deleting New Thesis



- We can delete the theses using “Delete” button.
- After deleting, the website is sent the warning message.

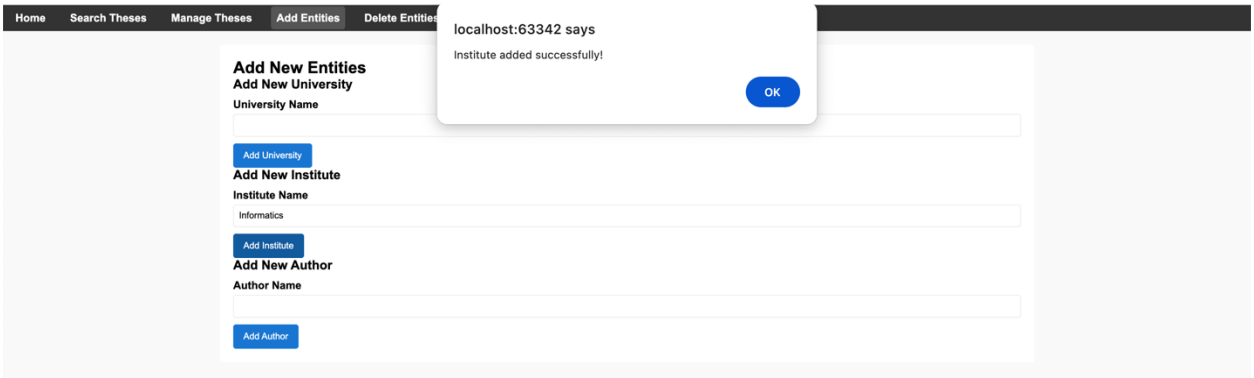
8.2.5 Add New Entities Page

8.2.5.1 Add New University



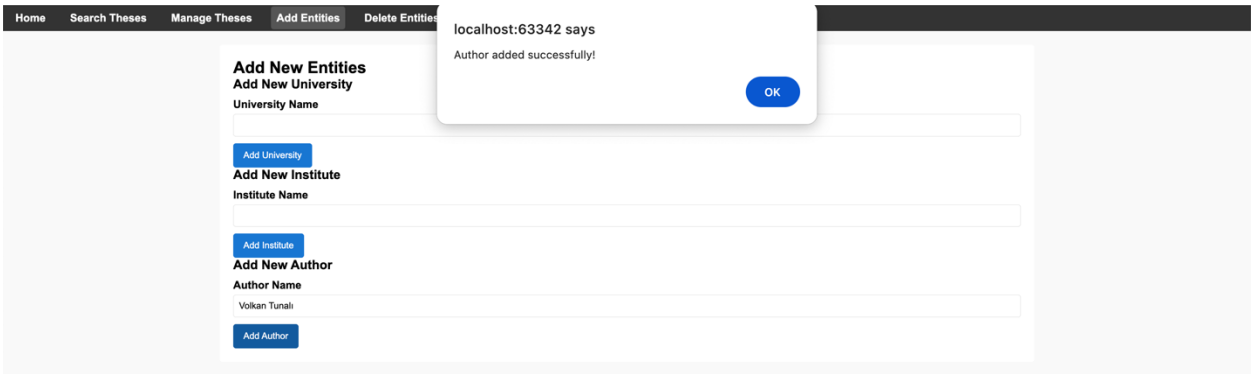
- When we enter the desired university in the University field and click the button, we receive a notification confirming that it has been added to the list.
- For example: Maltepe University

8.2.5.2 Add New Institute



- When we enter the desired institute in the Institute field and click the button, we receive a notification confirming that it has been added to the list.
- For example: Informatics

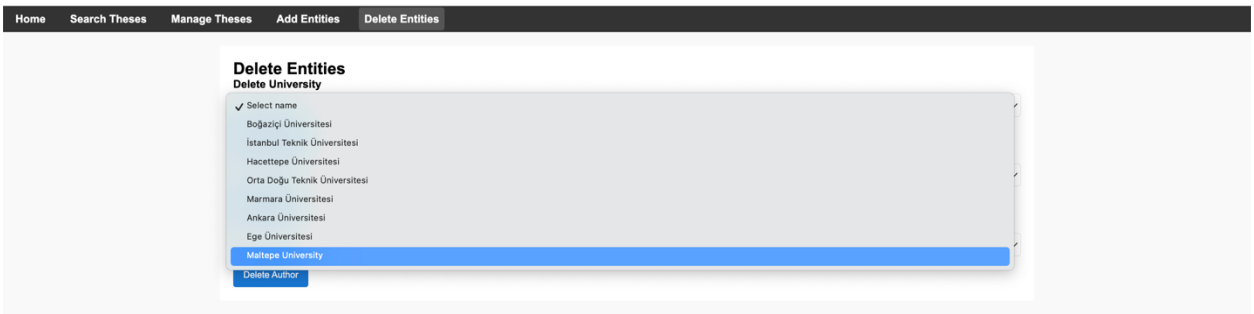
8.2.5.3 Add New Author



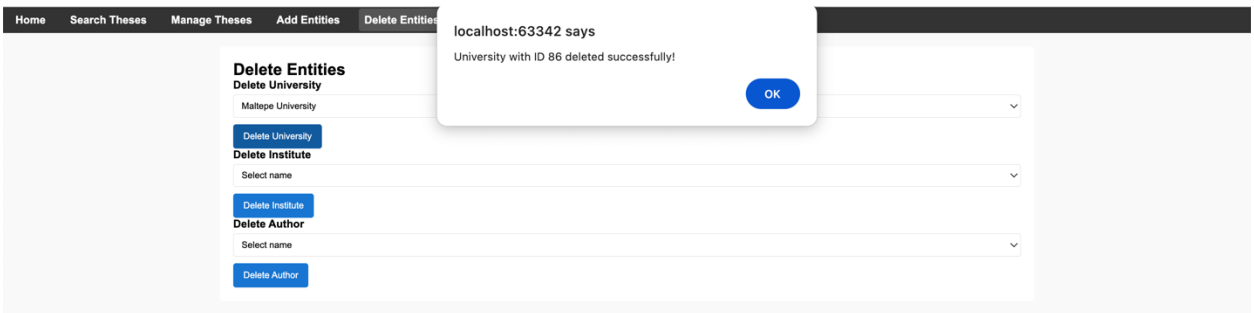
- When we enter the desired author in the Author field and click the button, we receive a notification confirming that it has been added to the list.
- For example: Volkan Tunalı

8.2.6 Delete Entities Page

8.2.6.1 Delete University

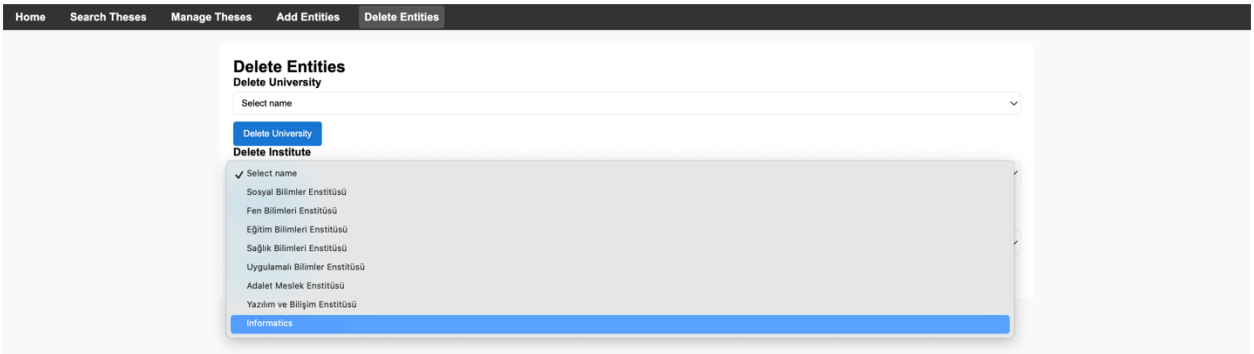


- In the previous part, we can see the variable we created listed, and we select it to delete it.

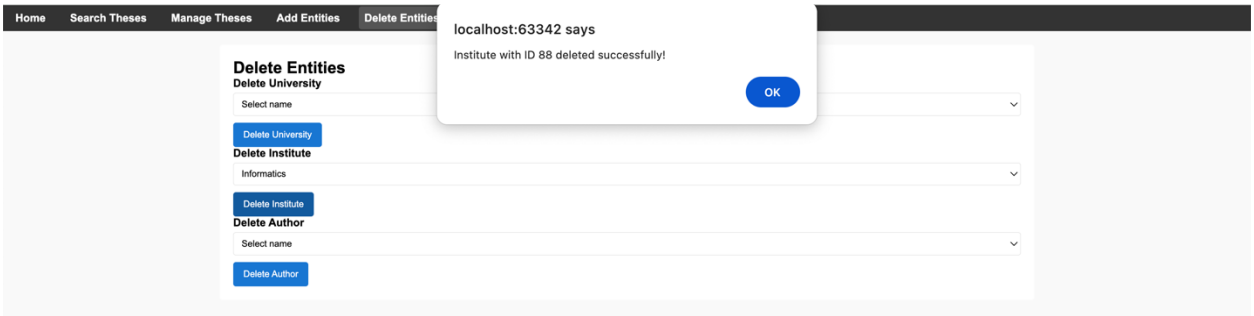


- It shows the details of the selected variable, confirming that the operation was successful.

8.2.6.2 Delete Institute

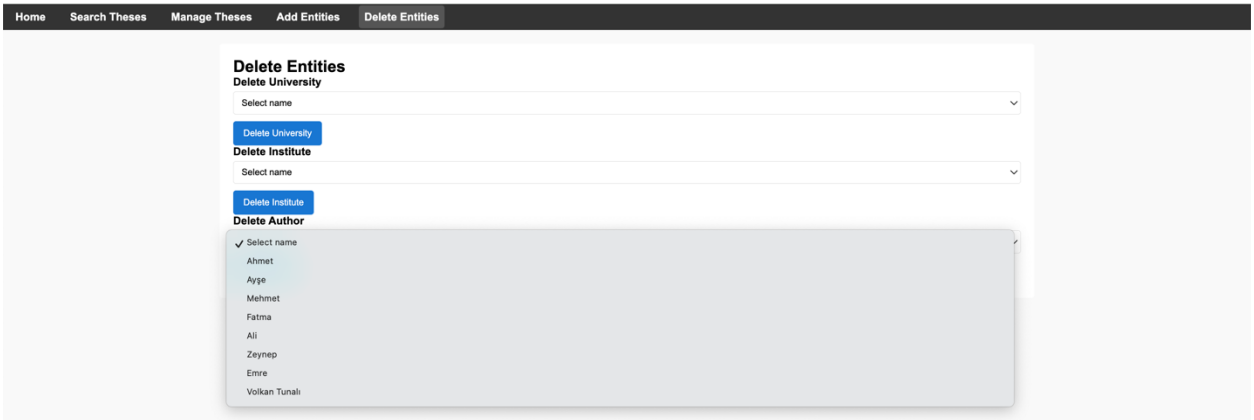


- In the previous part, we can see the variable we created listed, and we select it to delete it.

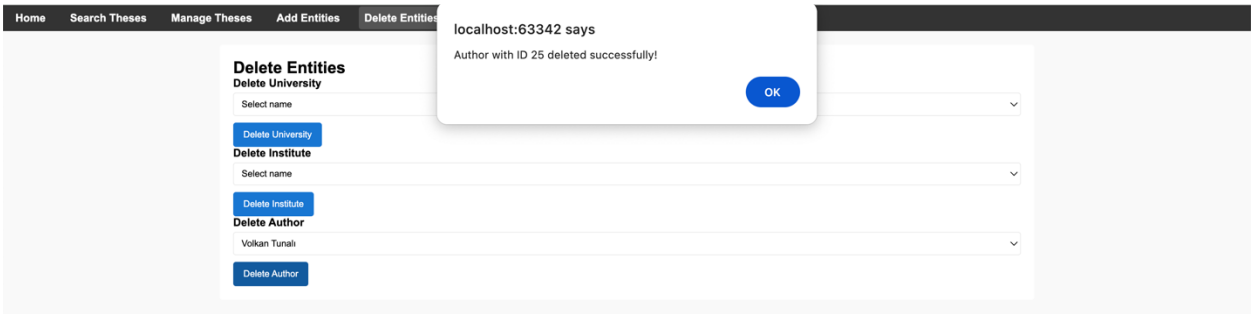


- It shows the details of the selected variable, confirming that the operation was successful.

8.2.6.3 Delete Author



- In the previous part, we can see the variable we created listed, and we select it to delete it.



- It shows the details of the selected variable, confirming that the operation was successful.

9. Database Access and Manipulation

9.1 Database Access

Our project uses **PostgreSQL** as the primary database, with Python's psycopg2 library for direct communication. A dedicated function, get_db_connection, establishes connections to the database:

```
import psycopg2

def get_db_connection():
    try:
        connection = psycopg2.connect(
            host="localhost",
            database="example_db",
            user="postgres",
            password="yourpassword"
        )
        return connection
    except psycopg2.Error as e:
        print(f"Database connection error: {e}")
        return None
```

9.2 Common Operations

Here's an example of retrieving data from the authors table:

```
@app.route('/authors', methods=['GET'])
def get_authors():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT author_id, first_name, last_name FROM
authors")
        authors = cursor.fetchall()
        cursor.close()
        conn.close()
        return jsonify([
            {"id": row[0], "first_name": row[1], "last_name": row[2]}
for row in authors
        ])
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

9.3 Challenges Encountered

Connection Management

- **Issue**: Handling database connections efficiently to avoid resource exhaustion.
- **Solution**: Use connection pooling with psycopg2.pool or a context manager for cleanup.

```
from psycopg2.pool import SimpleConnectionPool

connection_pool = SimpleConnectionPool(1, 10, ...)
```

SQL Injection

- **Issue**: Risk of SQL injection when using raw queries.
- **Solution**: Always use parameterized queries to pass user inputs securely:

```
cursor.execute("SELECT * FROM authors WHERE first_name = %s",
(first_name,))
```

Query Performance

- **Issue**: Slow performance for large datasets or complex joins.
- **Solution**: Use **indexes** for frequently queried columns:

```
CREATE INDEX idx_authors_lastname ON authors(last_name);
```

9.4 Techniques for Data Manipulation

9.4.1 Transactions

- **Purpose**: Ensure atomic operations.
- **Example**:

```
try:
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("INSERT INTO authors (first_name, last_name) VALUES
(%s, %s)", ("John", "Doe"))
```

```
    cursor.execute("INSERT INTO books (title, author_id) VALUES (%s,
%s)", ("Book Title", 1))
    conn.commit()
except Exception as e:
    conn.rollback()
    print(f"Transaction error: {e}")
```

9.4.2 Query Optimization

- **Purpose**: Improve performance for read-heavy operations.
- **Example**: Adding an index for faster lookups:

```
CREATE INDEX idx_authors_lastname ON authors(last_name);
```

9.4.3 Object Relational Mapping (ORM)

- **Purpose**: Simplify database interactions using Python objects.
- **Example**: Using SQLAlchemy:

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base

engine =
create_engine('postgresql://postgres:password@localhost/example_db')
Base = declarative_base()

class Author(Base):
    __tablename__ = 'authors'
    id = Column(Integer, primary_key=True)
    first_name = Column(String)
    last_name = Column(String)
```

9.5 Advantages of Our Approach

9.5.1 Scalability:

- Connection pooling ensures efficient use of resources.
- Query optimization techniques (e.g., indexing) enable fast performance for large datasets.

9.5.2 Data Integrity:

- Transactions ensure that operations succeed completely or fail safely.

- Foreign key constraints maintain relational consistency.

9.5.3 Security:

- Parameterized queries prevent SQL injection.

9.5.4 Ease of Use:

- The use of ORMs (e.g., SQLAlchemy) simplifies CRUD operations and reduces boilerplate code.

10. Appendix

10.1 Source Codes

10.1.1 Front-end Code

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Thesis Management - Home</title>
 <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
 <ul>
  <li><a id="home-link" href="index.html" class="active">Home</a></li>
  <li><a id="search-theses-link" href="search-theses.html">Search Theses</a></li>
  <li><a id="manage-theses-link" href="manage-theses.html">Manage Theses</a></li>
  <li><a id="add-entities-link" href="add-entities.html">Add Entities</a></li>
  <li><a id="delete-entities-link" href="delete-entities.html">Delete Entities</a></li>
  <li class="login-item"><a id="login-link" href="login.html">Login</a></li>
 </ul>
</nav>

<div class="page-container">
 <div class="intro-section card animated-fadeIn">
  <h2>How to Use This Project</h2>
  <p>
   This application is designed for managing academic theses. You can:
  </p>
  <ul>
   <li><strong>Search Theses:</strong> Go to the <em>Search Theses</em> page to find theses by author, keywords, year, or type.</li>
   <li><strong>Manage Theses:</strong> In the <em>Manage Theses</em> section, you can add new theses or update/delete existing ones. Dropdowns for authors, universities, and institutes are populated automatically.</li>
```

```
    <li><strong>Add Entities:</strong> Easily create new authors, universities, or institutes via <em>Add Entities</em>.</li>
    <li><strong>Delete Entities:</strong> Remove any existing authors, universities, or institutes using <em>Delete Entities</em>.</li>
    <li><strong>Login (Optional):</strong> If this project has user authentication enabled, you can log in to unlock additional features.</li>
   </ul>
   <p>
    Use the navigation bar at the top to switch between these sections. If the project requires authentication, some pages may be disabled until you log in.
   </p>
  </div>
  <br>
  <br>
  <br>
  <br>
  <div class="project-overview card animated-fadeIn">
   <h2>Project Overview</h2>
   <p>
    <strong>Technologies Used:</strong><br/>
    - <strong>Python (Flask):</strong> Our backend API is built using the Flask microframework, handling database operations and RESTful endpoints.<br/>
    - <strong>PostgreSQL:</strong> We store all thesis data, authors, institutes, and universities in a robust relational database.<br/>
    - <strong>HTML, CSS, JavaScript:</strong> The frontend is designed with semantic HTML, modern CSS (including transitions and basic animations), and vanilla JavaScript for dynamic interactions.<br/>
    - <strong>Fetch API:</strong> We use the Fetch API to send asynchronous requests to the Flask server, enabling operations like adding authors, deleting universities, searching theses, and more.<br/>
    - <strong>Bootstrap / or Plain CSS (Optional):</strong> You can integrate a CSS framework like Bootstrap for rapid UI development, or continue using custom CSS for a more tailored design.<br/>
   </p>
   <p>
    <strong>Key Features:</strong><br/>
    - <strong>Search Theses:</strong> Filter by author, keywords, year, or type.<br/>
    - <strong>Manage Theses:</strong> Create, update, and delete theses while automatically populating dropdowns (authors, universities, institutes).<br/>
    - <strong>Add &amp;amp; Delete Entities:</strong> Easily add or remove authors, universities, and institutes. PostgreSQL integrity ensures clean references.<br/>
    - <strong>Secure Login (Optional):</strong> A simple login system that demonstrates how to integrate user authentication with a separate <code>users</code> table.<br/>
   </p>
   <p>
    <br>
    <strong>How It Works:</strong><br/>
    - <em>Flask Backend:</em> We have multiple endpoints, such as <code>/add-thesis</code>, <code>/update-thesis/&lt;id&gt;</code>, <code>/delete-thesis/&lt;id&gt;</code>, etc.
    These endpoints handle creation, update, and deletion of records in the PostgreSQL database.<br/>
    - <em>Front-End:</em> Each page (<code>search-theses.html</code>, <code>manage-theses.html</code>, etc.) includes forms and tables.
    We use JavaScript (with the Fetch API) to talk to the Flask server, retrieving or sending JSON data.<br/>
```

```
    - <em>Database Schema:</em> The <code>thesis</code> table includes fields like
<code>title</code>, <code>abstract</code>, <code>year</code>, <code>type</code>, etc.,
    and references <code>authorid</code>, <code>universityid</code>, <code>instituteid</code>.
The <code>author</code>, <code>university</code>, and <code>institute</code>
    tables hold related info.
   </p>
   <p>
    This layout streamlines academic research and thesis management, making it simpler for you
    to explore, create, and maintain an organized database of theses.
    <em>Enjoy exploring the project and feel free to contribute!</em>
   </p>
  </div>
</div>
</body>
</html>
```

style.css

```
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}

body {
   font-family: Arial, sans-serif;
   background: #f9f9f9;
}

nav {
   background-color: #333;
   padding: 10px;
}

nav ul {
   list-style: none;
   display: flex;
   gap: 20px;
}

nav ul li {
   display: inline-block;
}

nav ul li a {
   color: #fff;
   text-decoration: none;
   font-weight: bold;
   padding: 8px;
}

nav ul li a.active,
nav ul li a:hover {
   background-color: #555;
```

```
    border-radius: 4px;
}

.page-container {
    max-width: 1200px;
    margin: 20px auto;
    background: #fff;
    padding: 20px;
    border-radius: 4px;
}

label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

input[type="text"],
input[type="number"],
input[type="date"],
select {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
}

button {
    padding: 10px 15px;
    margin-top: 10px;
    cursor: pointer;
    background-color: #1976d2;
    color: #fff;
    border: none;
    border-radius: 4px;
}

button:hover {
    background-color: #125a9e;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table,
th,
td {
    border: 1px solid #ddd;
}
```

```
th,
td {
  padding: 8px;
  text-align: left;
}

th {
  background-color: #f4f4f4;
}

nav ul {
  list-style: none;
  display: flex;
  gap: 20px;
  align-items: center;
  margin: 0;
  padding: 0;
}

.login-item {
  margin-left: auto;
}

.disabled {
  pointer-events: none;
  opacity: 0.5;
}
```

add-entities.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Add Entities</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="search-theses.html">Search Theses</a></li>
    <li><a href="manage-theses.html">Manage Theses</a></li>
    <li><a href="add-entities.html" class="active">Add Entities</a></li>
    <li><a href="delete-entities.html">Delete Entities</a></li>
  </ul>
</nav>

<div class="page-container">
  <h2>Add New Entities</h2>

  <div id="add-university-container">
```

```
    <h3>Add New University</h3>
    <form id="add-university-form">
      <label for="university-name">University Name</label>
      <input type="text" id="university-name" required />
      <button type="submit">Add University</button>
    </form>
  </div>

  <div id="add-institute-container">
    <h3>Add New Institute</h3>
    <form id="add-institute-form">
      <label for="institute-name">Institute Name</label>
      <input type="text" id="institute-name" required />
      <button type="submit">Add Institute</button>
    </form>
  </div>

  <div id="add-author-container">
    <h3>Add New Author</h3>
    <form id="add-author-form">
      <label for="author-name">Author Name</label>
      <input type="text" id="author-name" required />
      <button type="submit">Add Author</button>
    </form>
  </div>
</div>

<script src="script-common.js"></script>
<script src="script-add-entities.js"></script>
</body>
</html>
```

delete-entities.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Delete Entities</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="search-theses.html">Search Theses</a></li>
    <li><a href="manage-theses.html">Manage Theses</a></li>
    <li><a href="add-entities.html">Add Entities</a></li>
    <li><a href="delete-entities.html" class="active">Delete Entities</a></li>
  </ul>
</nav>

<div class="page-container">
```

```
  <h2>Delete Entities</h2>

  <div id="delete-university-container">
    <h4>Delete University</h4>
    <select id="delete-university-dropdown">
      <option value="">Select University</option>
    </select>
    <button id="delete-university-button">Delete University</button>
  </div>

  <div id="delete-institute-container">
    <h4>Delete Institute</h4>
    <select id="delete-institute-dropdown">
      <option value="">Select Institute</option>
    </select>
    <button id="delete-institute-button">Delete Institute</button>
  </div>

  <div id="delete-author-container">
    <h4>Delete Author</h4>
    <select id="delete-author-dropdown">
      <option value="">Select Author</option>
    </select>
    <button id="delete-author-button">Delete Author</button>
  </div>
</div>

<script src="script-common.js"></script>
<script src="script-delete-entities.js"></script>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Login</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
  <ul>
    <li><a id="home-link" href="index.html">Home</a></li>
    <li><a id="search-theses-link" href="search-theses.html">Search Theses</a></li>
    <li><a id="manage-theses-link" href="manage-theses.html">Manage Theses</a></li>
    <li><a id="add-entities-link" href="add-entities.html">Add Entities</a></li>
    <li><a id="delete-entities-link" href="delete-entities.html">Delete Entities</a></li>
    <li class="login-item"><a id="login-link" href="login.html" class="active">Login</a></li>
  </ul>
</nav>
```

```
<div class="page-container">
  <div class="card animated-fadeIn">
    <h2>Login</h2>
    <form id="login-form">
      <label for="username">Username</label>
      <input type="text" id="username" required />

      <label for="password">Password</label>
      <input type="password" id="password" required />

      <button type="submit" class="btn">Login</button>
    </form>
  </div>
</div>

<script>
  document.addEventListener("DOMContentLoaded", () => {
    const isLoggedIn = localStorage.getItem("loggedIn") === "true";
    const manageLink = document.getElementById("manage-theses-link");
    const addLink = document.getElementById("add-entities-link");
    const deleteLink = document.getElementById("delete-entities-link");

    if (!isLoggedIn) {
      manageLink.classList.add("disabled");
      addLink.classList.add("disabled");
      deleteLink.classList.add("disabled");
    } else {
      manageLink.classList.remove("disabled");
      addLink.classList.remove("disabled");
      deleteLink.classList.remove("disabled");
    }

    const loginForm = document.getElementById("login-form");
    loginForm.addEventListener("submit", (event) => {
      event.preventDefault();

      localStorage.setItem("loggedIn", "true");

      alert("Logged in successfully!");
      window.location.href = "index.html";
    });
  });
</script>
</body>
</html>
```

manage-theses.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```
  <title>Manage Theses</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="search-theses.html">Search Theses</a></li>
    <li><a href="manage-theses.html" class="active">Manage Theses</a></li>
    <li><a href="add-entities.html">Add Entities</a></li>
    <li><a href="delete-entities.html">Delete Entities</a></li>
  </ul>
</nav>

<div class="page-container">
  <h1>Thesis Management</h1>

  <div id="form-container">
    <h2 id="form-title">Add New Thesis</h2>
    <form id="add-thesis-form">
      <label for="title">Title</label>
            <input type="text" id="title" required />

      <label for="abstract">Abstract</label>
      <input type="text" id="abstract" />

      <label for="authorid">Author</label>
      <select id="authorid" required>
        <option value="">Select Author</option>
      </select>

      <label for="year">Year</label>
      <input type="number" id="year" required />

      <label for="type">Type</label>
      <select id="type" required>
        <option value="">Select Type</option>
        <option value="Master">Master</option>
        <option value="Doctorate">Doctorate</option>
        <option value="Specialization in Medicine">Specialization in Medicine</option>
      </select>

      <label for="universityid">University</label>
      <select id="universityid" required>
        <option value="">Select University</option>
      </select>

      <label for="instituteid">Institute</label>
      <select id="instituteid" required>
        <option value="">Select Institute</option>
      </select>

      <label for="numberofpages">Number of Pages</label>
```

```
        <input type="number" id="numberofpages" />

        <label for="language">Language</label>
        <select id="language" required>
          <option value="">Select Language</option>
          <option value="English">English</option>
          <option value="Turkish">Turkish</option>
          <option value="French">French</option>
          <option value="Spanish">Spanish</option>
          <option value="German">German</option>
          <option value="Chinese">Chinese</option>
          <option value="Japanese">Japanese</option>
          <option value="Russian">Russian</option>
          <option value="Arabic">Arabic</option>
          <option value="Portuguese">Portuguese</option>
          <option value="Italian">Italian</option>
          <option value="Hindi">Hindi</option>
          <option value="Korean">Korean</option>
        </select>

        <label for="submissiondate">Submission Date</label>
        <input type="date" id="submissiondate" required />

        <button type="submit" id="submit-button">Add Thesis</button>
      </form>
    </div>

    <h2>Theses List</h2>
    <table id="thesis-list">
      <thead>
      <tr>
        <th>ID</th>
        <th>Title</th>
        <th>Abstract</th>
        <th>Author</th>
        <th>Year</th>
        <th>Type</th>
        <th>University</th>
        <th>Institute</th>
        <th>Number of Pages</th>
        <th>Language</th>
        <th>Submission Date</th>
        <th>Actions</th>
      </tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>

  <script src="script-common.js"></script>
  <script src="script-manage-theses.js"></script>
  </body>
```

```
</html>
```

search-theses.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Search Theses</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="search-theses.html" class="active">Search Theses</a></li>
    <li><a href="manage-theses.html">Manage Theses</a></li>
    <li><a href="add-entities.html">Add Entities</a></li>
    <li><a href="delete-entities.html">Delete Entities</a></li>
  </ul>
</nav>

<div class="page-container">
  <h2>Search Theses</h2>
  <form id="search-theses-form">
    <label for="search-author">Author</label>
    <select id="search-author">
      <option value="">Select Author</option>
    </select>

    <label for="search-keywords">Keywords</label>
    <input type="text" id="search-keywords" placeholder="Enter keywords" />

    <label for="search-year">Year</label>
    <input type="number" id="search-year" placeholder="Enter year" />

    <label for="search-type">Type</label>
    <select id="search-type">
      <option value="">Select Type</option>
      <option value="Master">Master</option>
      <option value="Doctorate">Doctorate</option>
      <option value="Specialization in Medicine">Specialization in Medicine</option>
    </select>

    <button type="submit">Search</button>
  </form>

  <h2>Search Results</h2>
  <table id="thesis-list">
    <thead>
```

```
      <tr>
        <th>ID</th>
        <th>Title</th>
        <th>Abstract</th>
        <th>Author</th>
        <th>Year</th>
        <th>Type</th>
        <th>University</th>
        <th>Institute</th>
        <th>Number of Pages</th>
        <th>Language</th>
        <th>Submission Date</th>
        <th>Actions</th>
      </tr>
      </thead>
      <tbody>
      </tbody>
    </table>
</div>

<script src="script-common.js"></script>
<script src="script-search-theses.js"></script>
</body>
</html>
```

script.js

```
const apiUrl = "http://127.0.0.1:5000";

document.addEventListener("DOMContentLoaded", function () {
    loadDropdownsForDeletion();
    setupDeleteButtons();
    getTheses();
    loadDropdowns();
});

preventDuplicateEventListener("search-theses-form", function (event) {
    event.preventDefault();
    const author = document.getElementById("search-author").value;
    const keywords = document.getElementById("search-keywords").value.trim();
    const year = document.getElementById("search-year").value;
    const type = document.getElementById("search-type").value;

    fetch(`${apiUrl}/search-theses`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ author, keywords, year, type })
    })
        .then(response => {
            if (!response.ok) throw new Error("Failed to search theses");
            return response.json();
        })
        .then(data => {
            const tbody = document.querySelector("#thesis-list tbody");
```

```
        tbody.innerHTML = "";
        data.forEach(thesis => {
          const row = document.createElement("tr");
          row.innerHTML = `
      <td>${thesis.thesisid}</td>
      <td>${thesis.title}</td>
      <td>${thesis.abstract}</td>
      <td>${thesis.authorname || "Unknown Author"}</td>
      <td>${thesis.year}</td>
      <td>${thesis.type}</td>
      <td>${thesis.universityname || "Unknown University"}</td>
      <td>${thesis.institutename || "Unknown Institute"}</td>
      <td>${thesis.numberofpages || ""}</td>
      <td>${thesis.language}</td>
      <td>${thesis.submissiondate}</td>
      <td>
       <button onclick="editThesis(${thesis.thesisid})">Edit</button>
       <button onclick="deleteThesis(${thesis.thesisid})">Delete</button>
      </td>`;
          tbody.appendChild(row);
        });
      })
      .catch(error => console.error("Error searching theses:", error));
});

function getTheses() {
  fetch(`${apiUrl}/theses`)
      .then(response => {
        if (!response.ok) throw new Error("Failed to fetch theses");
        return response.json();
      })
      .then(data => {
        const tbody = document.querySelector("#thesis-list tbody");
        tbody.innerHTML = "";
        data.forEach(thesis => {
          const row = document.createElement("tr");
          row.innerHTML = `
       <td>${thesis.thesisid}</td>
       <td>${thesis.title}</td>
       <td>${thesis.abstract}</td>
       <td>${thesis.authorname || "Unknown Author"}</td>
       <td>${thesis.year}</td>
       <td>${thesis.type}</td>
       <td>${thesis.universityname || "Unknown University"}</td>
       <td>${thesis.institutename || "Unknown Institute"}</td>
       <td>${thesis.numberofpages || ""}</td>
       <td>${thesis.language}</td>
       <td>${thesis.submissiondate}</td>
       <td>
        <button onclick="editThesis(${thesis.thesisid})">Edit</button>
        <button onclick="deleteThesis(${thesis.thesisid})">Delete</button>
       </td>`;
          tbody.appendChild(row);
        });
```

```
    })
    .catch(error => console.error("Error fetching theses:", error));
}

function loadDropdowns() {
  fetchDropdown(`${apiUrl}/authors`, "authorid", "authorid", "name");
  fetchDropdown(`${apiUrl}/universities`, "universityid", "universityid", "name");
  fetchDropdown(`${apiUrl}/institutes`, "instituteid", "instituteid", "name");
  fetchDropdown(`${apiUrl}/authors`, "search-author", "authorid", "name");
}

function loadDropdownsForDeletion() {
  fetchDropdown(`${apiUrl}/authors`, "delete-author-dropdown", "authorid", "name");
  fetchDropdown(`${apiUrl}/universities`, "delete-university-dropdown", "universityid", "name");
  fetchDropdown(`${apiUrl}/institutes`, "delete-institute-dropdown", "instituteid", "name");
}

function fetchDropdown(url, elementId, valueKey, textKey) {
  fetch(url)
    .then(response => {
      if (!response.ok) throw new Error(`Failed to fetch data from ${url}`);
      return response.json();
    })
    .then(data => {
      const dropdown = document.getElementById(elementId);
      if (!dropdown) {
        console.error(`Dropdown with ID '${elementId}' not found.`);
        return;
      }
      dropdown.innerHTML = `<option value="">Select ${textKey}</option>`;
      data.forEach(item => {
        const option = document.createElement("option");
        option.value = item[valueKey];
        option.textContent = item[textKey];
        dropdown.appendChild(option);
      });
    })
    .catch(error => console.error(`Error fetching ${textKey}s:`, error));
}

preventDuplicateEventListener("add-author-form", function (event) {
  event.preventDefault();
  const authorName = document.getElementById("author-name").value.trim();
  if (!authorName) {
    alert("Author name cannot be empty!");
    return;
  }

  fetch(`${apiUrl}/add-author`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name: authorName })
  })
    .then(response => {
```

```
        if (!response.ok) throw new Error("Failed to add author");
        return response.json();
      })
      .then(data => {
        alert(data.message);
        document.getElementById("add-author-form").reset();
        loadDropdowns();
      })
      .catch(error => console.error("Error adding author:", error));
});

function setupDeleteButtons() {
  setupDeleteButton("delete-university-button", "delete-university-dropdown", "university");
  setupDeleteButton("delete-institute-button", "delete-institute-dropdown", "institute");
  setupDeleteButton("delete-author-button", "delete-author-dropdown", "author");
}

function setupDeleteButton(buttonId, dropdownId, type) {
  document.getElementById(buttonId).addEventListener("click", function () {
    const id = document.getElementById(dropdownId).value;
    if (!id) {
      alert(`Please select a ${type} to delete.`);
      return;
    }

    fetch(`${apiUrl}/delete-${type}/${id}`, { method: "DELETE" })
      .then(response => {
        if (!response.ok) throw new Error(`Failed to delete ${type}`);
        return response.json();
      })
      .then(data => {
        alert(data.message);
        loadDropdownsForDeletion();
      })
      .catch(error => console.error(`Error deleting ${type}:`, error));
  });
}

function preventDuplicateEventListener(formId, callback) {
  const form = document.getElementById(formId);
  if (!form.dataset.listenerAdded) {
    form.addEventListener("submit", callback);
    form.dataset.listenerAdded = "true";
  }
}
```

script-add-entities.js

```
document.addEventListener("DOMContentLoaded", () => {
  preventDuplicateEventListener("add-university-form", function (event) {
    event.preventDefault();
    const universityName = document.getElementById("university-name").value.trim();
    if (!universityName) {
      alert("University name cannot be empty!");
```

```
        return;
    }

    fetch(`${apiUrl}/add-university`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ name: universityName })
    })
        .then(response => {
            if (!response.ok) throw new Error("Failed to add university");
            return response.json();
        })
        .then(data => {
            alert(data.message);
            document.getElementById("add-university-form").reset();
        })
        .catch(error => console.error("Error adding university:", error));
});

preventDuplicateEventListener("add-institute-form", function (event) {
    event.preventDefault();
    const instituteName = document.getElementById("institute-name").value.trim();
    if (!instituteName) {
        alert("Institute name cannot be empty!");
        return;
    }

    fetch(`${apiUrl}/add-institute`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ name: instituteName })
    })
        .then(response => {
            if (!response.ok) throw new Error("Failed to add institute");
            return response.json();
        })
        .then(data => {
            alert(data.message);
            document.getElementById("add-institute-form").reset();
        })
        .catch(error => console.error("Error adding institute:", error));
});

preventDuplicateEventListener("add-author-form", function (event) {
    event.preventDefault();
    const authorName = document.getElementById("author-name").value.trim();
    if (!authorName) {
        alert("Author name cannot be empty!");
        return;
    }

    fetch(`${apiUrl}/add-author`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
```

```
        body: JSON.stringify({ name: authorName })
      })
        .then(response => {
          if (!response.ok) throw new Error("Failed to add author");
          return response.json();
        })
        .then(data => {
          alert(data.message);
          document.getElementById("add-author-form").reset();
        })
        .catch(error => console.error("Error adding author:", error));
  });
});
```

script-common.js

```
const apiUrl = "http://127.0.0.1:5000";

function fetchDropdown(url, elementId, valueKey, textKey) {
  fetch(url)
    .then(response => {
      if (!response.ok) throw new Error(`Failed to fetch data from ${url}`);
      return response.json();
    })
    .then(data => {
      const dropdown = document.getElementById(elementId);
      if (!dropdown) {
        console.error(`Dropdown with ID '${elementId}' not found.`);
        return;
      }
      dropdown.innerHTML = `<option value="">Select ${textKey}</option>`;
      data.forEach(item => {
        const option = document.createElement("option");
        option.value = item[valueKey];
        option.textContent = item[textKey];
        dropdown.appendChild(option);
      });
    })
    .catch(error => console.error(`Error fetching ${textKey}s:`, error));
}

function preventDuplicateEventListener(formId, callback) {
  const form = document.getElementById(formId);
  if (!form) return;
  if (!form.dataset.listenerAdded) {
    form.addEventListener("submit", callback);
    form.dataset.listenerAdded = "true";
  }
}

document.addEventListener("DOMContentLoaded", () => {
  const isLoggedIn = localStorage.getItem("loggedIn") === "true";
  const manageLink = document.getElementById("manage-theses-link");
  const addLink = document.getElementById("add-entities-link");
```

```
    const deleteLink = document.getElementById("delete-entities-link");

    if (!isLoggedIn) {
        manageLink.classList.add("disabled");
        addLink.classList.add("disabled");
        deleteLink.classList.add("disabled");
    } else {
        manageLink.classList.remove("disabled");
        addLink.classList.remove("disabled");
        deleteLink.classList.remove("disabled");
    }
});
```

script-delete-entities.js

```
document.addEventListener("DOMContentLoaded", () => {
    fetchDropdown(`${apiUrl}/authors`, "delete-author-dropdown", "authorid", "name");
    fetchDropdown(`${apiUrl}/universities`, "delete-university-dropdown", "universityid", "name");
    fetchDropdown(`${apiUrl}/institutes`, "delete-institute-dropdown", "instituteid", "name");

    setupDeleteButton("delete-university-button", "delete-university-dropdown", "university");
    setupDeleteButton("delete-institute-button", "delete-institute-dropdown", "institute");
    setupDeleteButton("delete-author-button", "delete-author-dropdown", "author");
});

function setupDeleteButton(buttonId, dropdownId, type) {
    const button = document.getElementById(buttonId);
    if (!button) return;

    button.addEventListener("click", () => {
        const id = document.getElementById(dropdownId).value;
        if (!id) {
            alert(`Please select a ${type} to delete.`);
            return;
        }

        fetch(`${apiUrl}/delete-${type}/${id}`, { method: "DELETE" })
            .then(response => {
                if (!response.ok) throw new Error(`Failed to delete ${type}`);
                return response.json();
            })
            .then(data => {
                alert(data.message);
                fetchDropdown(`${apiUrl}/authors`, "delete-author-dropdown", "authorid", "name");
                fetchDropdown(`${apiUrl}/universities`, "delete-university-dropdown", "universityid",
"name");
                fetchDropdown(`${apiUrl}/institutes`, "delete-institute-dropdown", "instituteid", "name");
            })
            .catch(error => console.error(`Error deleting ${type}:`, error));
    });
}
```

script-manage-entities.js

```
document.addEventListener("DOMContentLoaded", () => {
```

```
fetchDropdown(`${apiUrl}/authors`, "authorid", "authorid", "name");
fetchDropdown(`${apiUrl}/universities`, "universityid", "universityid", "name");
fetchDropdown(`${apiUrl}/institutes`, "instituteid", "instituteid", "name");

getTheses();

let editingThesisId = null;

const addThesisForm = document.getElementById("add-thesis-form");
const formTitle = document.getElementById("form-title");
const submitButton = document.getElementById("submit-button");

preventDuplicateEventListener("add-thesis-form", event => {
  event.preventDefault();

  const thesisData = {
    title: document.getElementById("title").value,
    abstract: document.getElementById("abstract").value,
    authorid: document.getElementById("authorid").value,
    year: document.getElementById("year").value,
    type: document.getElementById("type").value,
    universityid: document.getElementById("universityid").value,
    instituteid: document.getElementById("instituteid").value,
    numberofpages: document.getElementById("numberofpages").value,
    language: document.getElementById("language").value,
    submissiondate: document.getElementById("submissiondate").value
  };

  if (editingThesisId) {
    fetch(`${apiUrl}/update-thesis/${editingThesisId}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(thesisData)
    })
      .then(response => {
        if (!response.ok) throw new Error("Failed to update thesis");
        return response.json();
      })
      .then(data => {
        alert(data.message);
        addThesisForm.reset();
        editingThesisId = null;
        formTitle.textContent = "Add New Thesis";
        submitButton.textContent = "Add Thesis";
        getTheses();
      })
      .catch(error => console.error("Error updating thesis:", error));
  } else {
    fetch(`${apiUrl}/add-thesis`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(thesisData)
    })
      .then(response => {
```

```
        if (!response.ok) throw new Error("Failed to add thesis");
        return response.json();
      })
      .then(data => {
        alert(data.message);
        addThesisForm.reset();
        getTheses();
      })
      .catch(error => console.error("Error adding thesis:", error));
  }
});

function getTheses() {
  fetch(`${apiUrl}/theses`)
    .then(response => {
      if (!response.ok) throw new Error("Failed to fetch theses");
      return response.json();
    })
    .then(data => {
      const tbody = document.querySelector("#thesis-list tbody");
      if (!tbody) return;
      tbody.innerHTML = "";

      data.forEach(thesis => {
        const row = document.createElement("tr");
        row.innerHTML = `
          <td>${thesis.thesisid}</td>
          <td>${thesis.title}</td>
          <td>${thesis.abstract}</td>
          <td>${thesis.authorname || "Unknown Author"}</td>
          <td>${thesis.year}</td>
          <td>${thesis.type}</td>
          <td>${thesis.universityname || "Unknown University"}</td>
          <td>${thesis.institutename || "Unknown Institute"}</td>
          <td>${thesis.numberofpages || ""}</td>
          <td>${thesis.language}</td>
          <td>${thesis.submissiondate}</td>
          <td>
            <button onclick="editThesis(${thesis.thesisid})">Edit</button>
            <button onclick="deleteThesis(${thesis.thesisid})">Delete</button>
          </td>`;
        tbody.appendChild(row);
      });
    })
    .catch(error => console.error("Error fetching theses:", error));
}

window.editThesis = function (id) {
  editingThesisId = id;
  formTitle.textContent = "Edit Thesis (ID: " + id + ")";
  submitButton.textContent = "Update Thesis";

  alert(
    "Single-thesis endpoint is not implemented. You can manually fill the form and update."
```

```
    );
  };

  window.deleteThesis = function (id) {
    if (!confirm("Are you sure you want to delete this thesis?")) return;

    fetch(`${apiUrl}/delete-thesis/${id}`, {
      method: "DELETE"
    })
      .then(response => {
        if (!response.ok) throw new Error("Failed to delete thesis");
        return response.json();
      })
      .then(data => {
        alert(data.message);
        getTheses();
      })
      .catch(error => console.error("Error deleting thesis:", error));
  };
});
```

script-search-entities.js

```
document.addEventListener("DOMContentLoaded", () => {
  fetchDropdown(`${apiUrl}/authors`, "search-author", "authorid", "name");

  preventDuplicateEventListener("search-theses-form", event => {
    event.preventDefault();

    const author = document.getElementById("search-author").value;
    const keywords = document.getElementById("search-keywords").value.trim();
    const year = document.getElementById("search-year").value;
    const type = document.getElementById("search-type").value;

    fetch(`${apiUrl}/search-theses`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ author, keywords, year, type })
    })
      .then(response => {
        if (!response.ok) throw new Error("Failed to search theses");
        return response.json();
      })
      .then(data => {
        const tbody = document.querySelector("#thesis-list tbody");
        tbody.innerHTML = "";
        data.forEach(thesis => {
          const row = document.createElement("tr");
          row.innerHTML = `
            <td>${thesis.thesisid}</td>
            <td>${thesis.title}</td>
            <td>${thesis.abstract}</td>
            <td>${thesis.authorname || "Unknown Author"}</td>
            <td>${thesis.year}</td>
```

```
                <td>${thesis.type}</td>
                <td>${thesis.universityname || "Unknown University"}</td>
                <td>${thesis.institutename || "Unknown Institute"}</td>
                <td>${thesis.numberofpages || ""}</td>
                <td>${thesis.language}</td>
                <td>${thesis.submissiondate}</td>
                <td>
                  <button onclick="editThesis(${thesis.thesisid})">Edit</button>
                  <button onclick="deleteThesis(${thesis.thesisid})">Delete</button>
                </td>`;
              tbody.appendChild(row);
            });
          })
          .catch(error => console.error("Error searching theses:", error));
    });
});

function editThesis(id) {
    alert(`Edit Thesis with ID: ${id}`);
}

function deleteThesis(id) {
    alert(`Delete Thesis with ID: ${id}`);
}
```

10.1.2 Back-end Code

app.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import psycopg2

app = Flask(__name__)
CORS(app)

def get_db_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="SE307-TermProject",
            user="postgres",
            password="663732"
        )
        return conn
    except Exception as e:
        print(f"Error connecting to database: {e}")
        return None

@app.route('/add-author', methods=['POST'])
def add_author():
    try:
        data = request.get_json()
        name = data.get('name')
```

```
        if not name:
            return jsonify({"error": "Author name is required"}), 400

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO author (name) VALUES (%s)", (name,))
        conn.commit()
        cursor.close()
        conn.close()

        return jsonify({"message": "Author added successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/add-institute', methods=['POST'])
def add_institute():
    try:
        data = request.get_json()
        name = data.get('name')
        if not name:
            return jsonify({"error": "Institute name is required"}), 400

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO institute (name) VALUES (%s)", (name,))
        conn.commit()
        cursor.close()
        conn.close()

        return jsonify({"message": "Institute added successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/add-university', methods=['POST'])
def add_university():
    try:
        data = request.get_json()
        name = data.get('name')

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO university (name) VALUES (%s)", (name,))
        conn.commit()
        cursor.close()
        conn.close()

        return jsonify({"message": "University added successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/theses', methods=['GET'])
def get_theses():
    try:
        conn = get_db_connection()
```

```
    cursor = conn.cursor()
    cursor.execute("""
      SELECT
        t.thesisid, t.title, t.abstract, a.name AS authorname, t.year, t.type,
        u.name AS universityname, i.name AS institutename, t.numberofpages,
        t.language, t.submissiondate
      FROM
        thesis t
      LEFT JOIN author a ON t.authorid = a.authorid
      LEFT JOIN university u ON t.universityid = u.universityid
      LEFT JOIN institute i ON t.instituteid = i.instituteid
    """)
    theses = cursor.fetchall()
    cursor.close()
    conn.close()

    result = [
      {
        "thesisid": row[0],
        "title": row[1],
        "abstract": row[2],
        "authorname": row[3],
        "year": row[4],
        "type": row[5],
        "universityname": row[6],
        "institutename": row[7],
        "numberofpages": row[8],
        "language": row[9],
        "submissiondate": row[10]
      } for row in theses
    ]
    return jsonify(result)
  except Exception as e:
    return jsonify({"error": str(e)}), 500

@app.route('/add-thesis', methods=['POST'])
def add_thesis():
  try:
    data = request.get_json()
    allowed_languages = [
      "English", "Turkish", "French", "Spanish", "German", "Chinese",
      "Japanese", "Russian", "Arabic", "Portuguese", "Italian", "Hindi", "Korean"
    ]

    if data['language'] not in allowed_languages:
      return jsonify({"error": f"Invalid language. Allowed values are: {',
'.join(allowed_languages)}"}), 400

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
      INSERT INTO thesis (title, abstract, authorid, year, type, universityid, instituteid,
numberofpages, language, submissiondate)
      VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
```

```
        """, (
            data['title'], data.get('abstract', ''), data['authorid'], data['year'], data['type'],
            data['universityid'], data['instituteid'], data.get('numberofpages', None),
            data['language'], data['submissiondate']
        ))
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({"message": "Thesis added successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/update-thesis/<int:thesisid>', methods=['PUT'])
def update_thesis(thesisid):
    try:
        data = request.get_json()
        allowed_languages = [
            "English", "Turkish", "French", "Spanish", "German", "Chinese",
            "Japanese", "Russian", "Arabic", "Portuguese", "Italian", "Hindi", "Korean"
        ]

        if data['language'] not in allowed_languages:
            return jsonify({"error": f"Invalid language. Allowed values are: {',
'.join(allowed_languages)}"}), 400

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("""
            UPDATE thesis
            SET title = %s, abstract = %s, authorid = %s, year = %s, type = %s,
                universityid = %s, instituteid = %s, numberofpages = %s, language = %s, submissiondate =
%s
            WHERE thesisid = %s
        """, (
            data['title'], data.get('abstract', ''), data['authorid'], data['year'], data['type'],
            data['universityid'], data['instituteid'], data.get('numberofpages', None),
            data['language'], data['submissiondate'], thesisid
        ))
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({"message": f"Thesis with ID {thesisid} updated successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/delete-thesis/<int:thesisid>', methods=['DELETE'])
def delete_thesis(thesisid):
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("DELETE FROM thesissubject WHERE thesisid = %s", (thesisid,))
        cursor.execute("DELETE FROM thesiskeyword WHERE thesisid = %s", (thesisid,))
        cursor.execute("DELETE FROM thesissupervisor WHERE thesisid = %s", (thesisid,))
        cursor.execute("DELETE FROM thesis WHERE thesisid = %s", (thesisid,))
```

```
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({"message": f"Thesis with ID {thesisid} deleted successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/authors', methods=['GET'])
def get_authors():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT authorid, name FROM author")
        authors = cursor.fetchall()
        cursor.close()
        conn.close()

        result = [{"authorid": author[0], "name": author[1]} for author in authors]
        return jsonify(result)
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/universities', methods=['GET'])
def get_universities():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT universityid, name FROM university")
        universities = cursor.fetchall()
        cursor.close()
        conn.close()

        result = [{"universityid": university[0], "name": university[1]} for university in universities]
        return jsonify(result)
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/delete-university/<int:universityid>', methods=['DELETE'])
def delete_university(universityid):
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("DELETE FROM university WHERE universityid = %s", (universityid,))
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({"message": f"University with ID {universityid} deleted successfully!"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

@app.route('/delete-institute/<int:instituteid>', methods=['DELETE'])
def delete_institute(instituteid):
    try:
        conn = get_db_connection()
```

```
    cursor = conn.cursor()
    cursor.execute("DELETE FROM institute WHERE instituteid = %s", (instituteid,))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": f"Institute with ID {instituteid} deleted successfully!"})
  except Exception as e:
    return jsonify({"error": str(e)}), 500

@app.route('/delete-author/<int:authorid>', methods=['DELETE'])
def delete_author(authorid):
  try:
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM author WHERE authorid = %s", (authorid,))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": f"Author with ID {authorid} deleted successfully!"})
  except Exception as e:
    return jsonify({"error": str(e)}), 500

@app.route('/search-theses', methods=['POST'])
def search_theses():
  try:
    data = request.get_json()
    author = data.get('author')
    keywords = data.get('keywords')
    year = data.get('year')
    thesis_type = data.get('type')

    query = """
      SELECT
        t.thesisid, t.title, t.abstract, a.name AS authorname, t.year, t.type,
        u.name AS universityname, i.name AS institutename, t.numberofpages,
        t.language, t.submissiondate
      FROM
        thesis t
      LEFT JOIN author a ON t.authorid = a.authorid
      LEFT JOIN university u ON t.universityid = u.universityid
      LEFT JOIN institute i ON t.instituteid = i.instituteid
      WHERE 1=1
    """

    params = []
    if author:
      query += " AND t.authorid = %s"
      params.append(author)
    if keywords:
      query += " AND (t.title ILIKE %s OR t.abstract ILIKE %s)"
      keyword_filter = f"%{keywords}%"
      params.extend([keyword_filter, keyword_filter])
    if year:
      query += " AND t.year = %s"
```

```
        params.append(year)
    if thesis_type:
        query += " AND t.type = %s"
        params.append(thesis_type)

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute(query, tuple(params))
    theses = cursor.fetchall()
    cursor.close()
    conn.close()

    result = [
        {
            "thesisid": row[0],
            "title": row[1],
            "abstract": row[2],
            "authorname": row[3],
            "year": row[4],
            "type": row[5],
            "universityname": row[6],
            "institutename": row[7],
            "numberofpages": row[8],
            "language": row[9],
            "submissiondate": row[10]
        } for row in theses
    ]

    return jsonify(result)
  except Exception as e:
    return jsonify({"error": str(e)}), 500

@app.route('/institutes', methods=['GET'])
def get_institutes():
  try:
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT instituteid, name FROM institute")
    institutes = cursor.fetchall()
    cursor.close()
    conn.close()

    result = [{"instituteid": institute[0], "name": institute[1]} for institute in institutes]
    return jsonify(result)
  except Exception as e:
    return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
  app.run(debug=True)
```

10.2 Self-Reflection

Muhammed Numan Karaca:

The project was actually very productive; it was based on our efforts and learning. In fact, it allowed us to apply the topics covered in class, and as a result, it became one of the most realistic projects we've ever worked on, requiring serious thought and effort. I struggled a bit with some steps because I think I might have missed some points during the lectures, which could be why I found it challenging. However, as a team, we believe we successfully completed the application and documentation stages, even if it required extensive effort and time.

However, while working on the project, I not only utilized some of my existing knowledge but also gained significant new insights. Sticking to the defined schedule and learning about certain technologies were among the important achievements I obtained.

This experience showed us that in the industry, coding alone doesn't hold all the significance—documentation processes are equally important. It was a satisfying experience to document our work, even in a simple format, and to detail everything to the smallest extent possible.

Thank you so much for everything up to this point. Your efforts are greatly appreciated, Volkan Hocam.

Ömer Faruk Özer:

The purpose and scope of the project were quite intriguing. During this process, I learned how to integrate backend and frontend technologies, gaining a better understanding of the dynamics between the client and server sides. Additionally, exploring and effectively applying methods to make changes in the database was a significant achievement for me. Working with PostgreSQL, in particular, helped me enhance my skills in data management.

However, my teammates and I encountered various challenges throughout the project. The broad scope and heavy workload of the project occasionally caused issues in team communication and task distribution. Nevertheless, these difficulties helped me better understand the importance of teamwork and resilience.

Throughout the process, I not only improved my technical skills but also gained valuable experience in time management, problem-solving, and team coordination. The lessons I learned while completing this project will serve as a valuable guide for my future endeavors.

Berke Ensel:

I believe this was the most important course we took this year. Since I plan to focus on database management, data science, and artificial intelligence in the future, this course was especially significant for me. Beyond just the academic aspects, I genuinely felt that you tried to provide us with valuable insights and practical knowledge. It was clear that you care about your students and aim to guide them with a solid road map.
Before taking this course, I had some prior knowledge of databases and SQL, but this class allowed me to build upon that foundation and significantly improve my skills. The way the course was delivered and the progression of the topics were excellent.
Thank you so much for everything Volkan Hocam.