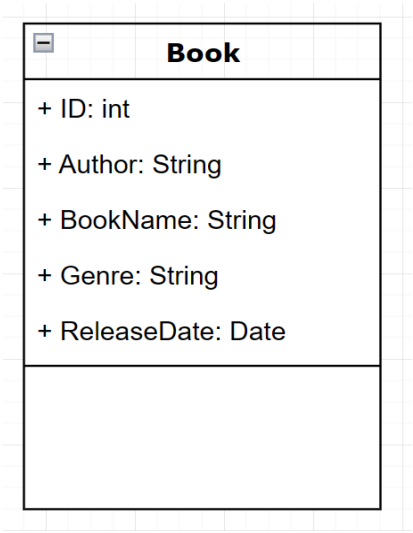
Library Management

Sinan Keskin 1904385
Furkan Vural Okur 1904569

Project Aim: Our aim in our library management system is to store books and information about books in one place and to be able to view them easily. In addition, we can manage operations such as adding books, deleting books and editing existing book information through the application.

Member-task responsibilities:
Sinan Keskin %50 Data Entry (Input Form) , Result Page, %50 Database Tasks
Furkan Vural Okur %50 Data Entry (Input Form), %50 Database Tasks, Internationalization



Controller :

A data transfer object is sent in the book adding part and a record is recorded in the database by matching with the appropriate data in the "Book" Entity.

In the List Book section, all available books in the database are displayed. If the list is empty, the Empty page opens.

In the Edit section, a data transfer object is sent as in the book addition section, but the existing record is edited as soon as a new record is made.

In Delete, the books are found according to their id, whichever book we choose is deleted.
Code Block:

```
@Controller
@RequestMapping("")
public class BookController {
   private final SessionLocaleResolver localeResolver;

   public BookController(SessionLocaleResolver localeResolver) {
      this.localeResolver = localeResolver;
   }

   @PostMapping("/change-language")
   public String changeLanguage(@RequestParam("lang") String language, HttpServletRequest request, HttpServletResponse
response) {
      Locale locale = new Locale(language);
      localeResolver.setLocale(request, response, locale);
      return "redirect:/"; // Redirect to the desired page
   }
```

```
    @Autowired
    private BookRepository bookRepository;

    @GetMapping("/add-book")
    public String showAddBookForm(Model model) {
        model.addAttribute("book", new Book());
        return "add-book";
    }

    @PostMapping("/add-book")
    public String addBook(@ModelAttribute("book") BookDto book) {
        Book bookAdd = new Book();
        bookAdd.setBookname(book.getBookname());
        bookAdd.setGenre(book.getGenre());
        bookAdd.setAuthor(book.getAuthor());
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm");
        LocalDateTime date = LocalDateTime.parse(book.getYear(), formatter);
        bookAdd.setYear(date);
        bookRepository.save(bookAdd);
        return "redirect:/list-books";
    }

    @GetMapping("/list-books")
    public String listBooks(Model model) {
        List<Book> books = bookRepository.findAll();
        if(bookRepository.count() != 0) {
            model.addAttribute("books", books);
            return "list_books";
        }
        else {
            return "redirect:/empty";
        }
    }
    @GetMapping("/empty")
    public String emptyPage(){
        return "empty";
    }

    @GetMapping("/edit-book/{id}")
    public String showEditBookForm(@PathVariable("id") Long id, Model model) {
        Book book = bookRepository.findById(id)
                .orElseThrow(() -> new IllegalArgumentException("Invalid book Id: " + id));
        model.addAttribute("book", book);
        return "edit-book";
    }

    @PostMapping("/edit-book/{id}")
    public String editBook(@PathVariable("id") Long id, @ModelAttribute("book") BookDto bookDetails) {
        Book book = bookRepository.findById(id)
                .orElseThrow(() -> new IllegalArgumentException("Invalid book Id: " + id));
        book.setBookname(bookDetails.getBookname());
        book.setAuthor(bookDetails.getAuthor());
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm");
        LocalDateTime date = LocalDateTime.parse(bookDetails.getYear(), formatter);
        book.setYear(date);
        book.setGenre(bookDetails.getGenre());
        bookRepository.save(book);
        return "redirect:/list-books";
    }

    @PostMapping("/delete-book/{id}")
    public String deleteBook(@PathVariable("id") Long id) {
        Book book = bookRepository.findById(id)
                .orElseThrow(() -> new IllegalArgumentException("Invalid book Id: " + id));
        bookRepository.deleteById(book.getId());
        return "redirect:/list-books";
    }
}
```

AppConfig :

It is a java file that contains the necessary configurations for both i18n and thymeleaf.

```
@Configuration
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware, EnvironmentAware {

    private ApplicationContext applicationContext;
    private Environment environment;

    @Bean
    public SessionLocaleResolver sessionLocaleResolver() {
        SessionLocaleResolver resolver = new SessionLocaleResolver();
        resolver.setDefaultLocale(Locale.ENGLISH);
        return resolver;
    }
    @Bean
    public LocaleResolver localeResolver1() {
        return new SessionLocaleResolver();
    }

    @Bean
    public LocaleResolver localeResolver() {
        CookieLocaleResolver resolver = new CookieLocaleResolver();
        resolver.setDefaultLocale(Locale.ENGLISH);
        resolver.setCookieName("localeCookie");
        resolver.setCookieMaxAge(3600);
        return resolver;
    }

    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSource();
        messageSource.setBasename("classpath:/");
        messageSource.setDefaultEncoding("UTF-8");
        messageSource.setCacheSeconds(0);
        return messageSource;
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry){
        LocaleChangeInterceptor interceptor =new LocaleChangeInterceptor();
        interceptor.setParamName("language");
        registry.addInterceptor(interceptor);
    }

    public void setApplicationContext(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver resolver = new SpringResourceTemplateResolver();
        resolver.setApplicationContext(applicationContext);
        resolver.setPrefix("classpath:/templates/");
        resolver.setSuffix(".html");
        resolver.setTemplateMode(TemplateMode.HTML);
        resolver.setCharacterEncoding("UTF-8"); // Set the character encoding
        return resolver;
    }


    @Override
    public void setEnvironment(Environment environment) {
```

```
        this.environment = environment;
    }
}
```

BookDto :

It contains the data that the user sends to the system in the book adding and editing sections.

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class BookDto {
    private Long id;
    private String bookname;
    private String author;
    private String genre;
    private String year;
}
```

Book :

It is the file containing the data types to be saved in the table in the database and shown in the listing section. Data in Data transfer object must conform to the data formats in this file when saving to the database.

```
@Entity
@Data
@Table(name = "BOOK")
public class Book implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name="BOOKNAME")
    private String bookname;
    @Column(name="AUTHOR")
    private String author;
    @Column(name="GENRE")
    private String genre;
    @Column(name = "PUBLISH_DATE")
    private LocalDateTime year;
```

BookRepository :

The book repository is used when performing CRUD operations.

```
@Repository
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

BookService:

Operations such as adding and deleting data used in the controller are managed by the service.

```
@Service
@Slf4j
@RequiredArgsConstructor
public class BookService {

    private final BookRepository bookRepository;


    public Book saveBook(Book book) { return bookRepository.save(book); }

    public void deleteBook(Long bookId) {
        bookRepository.deleteById(bookId);
    }
}
```

LibraryApplication:

This is where the Spring Boot application runs. The configuration file is imported (if any).

```
@SpringBootApplication
@Import(AppConfig.class)
public class LibraryApplication {

	public static void main(String[] args) {
		SpringApplication.run(LibraryApplication.class, args);
	}

}
```

add-book.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Add Book</title>
  <style>
  form {
        margin: 0 auto;
        text-align: center;
        width: 30%;
        margin-left:33%;
      }
    .form-row {
        display: flex;
        align-items: center;
        margin-bottom: 10px;
      }

      .form-row label {
        flex: 0 0 100px;
        margin-right: 10px;
      }

      .form-row input {
        flex: 1;
      }

      h1, label,input,p {
      text-align:center;
      }
      button {
      margin-left: 12%;
      }

  </style>
</head>
<body>
<h1>Add Book</h1>
<form th:action="@{/add-book}" th:object="${book}" method="post">
  <div class="form-row">
  <label for="bookname" >Title:</label>
  <input type="text" id="bookname" name="bookname" th:field="*{bookname}"required><br><br>
  </div>
  <div class="form-row">
  <label for="author" >Author:</label>
  <input type="text" id="author" name="author" th:field="*{author}" required><br><br>
  </div>
  <div class="form-row">
  <label for="author">Genre:</label>
```

```
  <input type="text" id="genre" name="genre" th:field="*{genre}" required><br><br>
  </div>
  <div class="form-row">
  <label for="year" >Year:</label>
  <input type="datetime-local" id="year" name="year" th:field="*{year}" required><br><br>
  </div>
  <button type="submit">Add Book</button>
</form>
<p><a href="?language=tr">Türkçe</a> | <a href="?language=en">English</a> </p>
</body>
</html>
```

edit-book.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" th:lang="${#locale.language}">
<head>
    <meta charset="UTF-8">
    <title>Edit Book</title>
    <style>
        form {
            margin: 0 auto;
            text-align: center;
            width: 30%;
            margin-left:33%;
        }
    .form-row {
            display: flex;
            align-items: center;
            margin-bottom: 10px;
        }

        .form-row label {
            flex: 0 0 100px;
            margin-right: 10px;
        }

        .form-row input {
            flex: 1;
        }

        h1, label,input,p {
        text-align:center;
        }
        button {
        margin-left: 12%;
        }
    </style>
</head>
<body>
<h1>Edit Book</h1>
<form th:action="@{/edit-book/{id}(id=${book.id})}" th:object="${book}" method="post">
    <div class="form-row">
    <label for="bookname" >Title:</label>
    <input type="text" id="bookname" name="bookname" th:field="*{bookname}"required><br><br>
    </div>
    <div class="form-row">
    <label for="bookname" >Title:</label>
    <input type="text" id="author" name="author" th:field="*{author}" required><br><br>
    </div>
    <div class="form-row">
    <label for="genre">Genre:</label>
    <input type="text" id="genre" name="genre" th:field="*{genre}" required><br><br>
    </div>
    <div class="form-row">
    <label for="datetime" >Publish Date:</label>
    <input type="datetime-local" id="datetime" name="datetime" th:field="*{year}" required><br><br>
    </div>
```

```
    <button type="submit" >Edit Book</button>
</form>
<p><a href="?language=tr">Türkçe</a> | <a href="?language=en">English</a> </p>

</body>
</html>
```

empty.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" th:lang="${#locale.language}">
<head>
    <meta charset="UTF-8">
    <title th:text="#{nodata.title}">No data at database!</title>
</head>
<body>
<p >Looks like there isn't any book in our database.</p>
<form action="/add-book" method="get">
    <button type="submit" >Let's add a new one</button>
</form>
<p><a href="?language=tr">Türkçe</a> | <a href="?language=en">English</a> </p>

</body>
</html>
```

list_books.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" th:lang="${#locale.language}">
<head>
    <meta charset="UTF-8">
    <title>List Books</title>
    <style>
        table {
            margin: 0 auto;
            text-align: center;
            border-collapse: collapse;
            width: 100%;
        }

        th, td {
            border: 1px solid black; /* Add borders to table cells */
            padding: 8px; /* Add padding inside cells */
        }

        h1, p {
        text-align:center;
        }

        .button {
            display: inline-block;
            padding: 10px 20px;
            text-align: center;
            text-decoration: none;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            margin-left: 46%;
            margin-top: 1%;
        }
    </style>
</head>
<body>
<h1 >List Books</h1>
```

```
<table>
  <tr>
    <th>ID</th>
    <th>Title</th>
    <th >Author</th>
    <th >Genre</th>
    <th >Date</th>
    <th >Edit</th>
    <th >Delete</th>
  </tr>
  <tr th:each="book : ${books}">
    <td th:text="${book.id}"></td>
    <td th:text="${book.bookname}"></td>
    <td th:text="${book.author}"></td>
    <td th:text="${book.genre}"></td>
    <td th:text="${#temporals.format(book.year, 'dd-MM-yyyy')}"></td>
    <td>
      <a th:href="@{/edit-book/{id}(id=${book.id})}" >Edit</a>
    </td>
    <td>
      <form th:action="@{/delete-book/{id}(id=${book.id})}" method="post">
        <button type="submit" >Delete</button>
      </form>
    </td>
  </tr>
</table>
<a href="/add-book" class="button">Add New Book</a>
<p><a href="?language=tr">Türkçe</a> | <a href="?language=en">English</a> </p>

</body>
</html>
```

application.properties :

Just like AppConfig, it is the file that contains the configurations (database, datasource, entity manager, etc.) that will be required while the application is running.

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.h2.console.settings.trace=false
spring.h2.console.settings.web-allow-others=false
spring.jpa.hibernate.ddl-auto=update
spring.datasource.initialization-mode=always
spring.datasource.data=classpath:data.sql
spring.datasource.schema=classpath:schema.sql
spring.thymeleaf.enabled=true
spring.messages.basename=messages
spring.messages.encoding=UTF-8
spring.main.allow-circular-references=true

# Thymeleaf configuration
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
spring.thymeleaf.encoding=UTF-8
spring.thymeleaf.mode=HTML
spring.thymeleaf.cache=false
spring.main.allow-bean-definition-overriding=true
```

data.sql :

While the application is running, data insertion is done through data.sql.

```
INSERT INTO "BOOK" ("ID", "AUTHOR", "BOOKNAME", "GENRE", "PUBLISH_DATE")
VALUES (0, 'John Smith', 'The Book', 'Fiction', '2023-01-01 10:00:00');
```

schema.sql :

It contains the tables that will be created when the application runs.

```
CREATE TABLE "BOOK" (
  "ID" BIGINT GENERATED BY DEFAULT AS IDENTITY,
  "AUTHOR" VARCHAR(255),
  "BOOKNAME" VARCHAR(255),
  "GENRE" VARCHAR(255),
  "PUBLISH_DATE" DATETIME ,
  PRIMARY KEY ("ID")
);
```

messages.properties :

Contains articles for i18n.

```
addbook.title=Add Book
addbook.heading=Add Book
addbook.label.title=Title:
addbook.label.author=Author:
addbook.label.genre=Genre:
addbook.label.year=Year:
addbook.button=Add Book
editbook.title=Edit Book
editbook.heading=Edit Book
editbook.label.title=Title:
editbook.label.author=Author:
editbook.label.genre=Genre:
editbook.label.date=Publish Date:
editbook.button=Edit Book
nodata.title=No data at database!
nodata.message=Looks like there isn't any book in our database.
nodata.button=Let's add a new one
listbooks.title=List Books
listbooks.heading=List Books
listbooks.column.title=Title
listbooks.column.author=Author
listbooks.column.genre=Genre
listbooks.column.date=Date
listbooks.column.edit=Edit
listbooks.column.delete=Delete
listbooks.button.edit=Edit
listbooks.button.delete=Delete
```

messages_tr.properties :

```
addbook.title=Kitap Ekle
addbook.heading=Kitap Ekle
addbook.label.title=Ba?l?k:
addbook.label.author=Yazar:
addbook.label.genre=T�r:
addbook.label.year=Y?l:
addbook.button=Kitap Ekle
editbook.title=Kitab? D�zenle
editbook.heading=Kitab? D�zenle
editbook.label.title=Ba?l?k:
editbook.label.author=Yazar:
editbook.label.genre=T�r:
editbook.label.date=Yay?n Tarihi:
editbook.button=Kitab? D�zenle
nodata.title=Veritaban?nda veri bulunamad?!
nodata.message=Veritaban?m?zda herhangi bir kitap bulunmuyor gibi g�r�n�yor.
```

```
nodata.button=Yeni bir tane ekleyelim
listbooks.title=Kitap Listesi
listbooks.heading=Kitap Listesi
listbooks.column.title=Ba?l?k
listbooks.column.author=Yazar
listbooks.column.genre=T�r
listbooks.column.date=Tarih
listbooks.column.edit=D�zenle
listbooks.column.delete=Sil
listbooks.button.edit=D�zenle
listbooks.button.delete=Sil
```

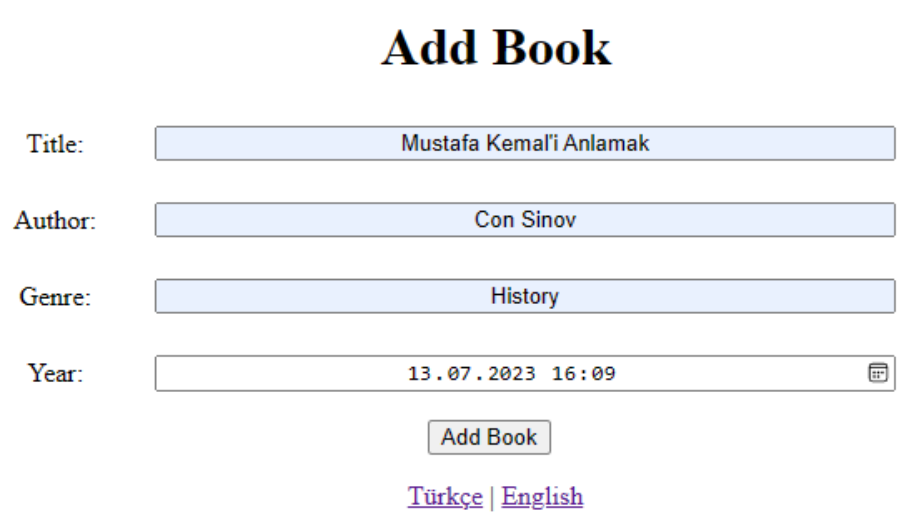
pom.xml :

It contains all the dependencies used in the app.

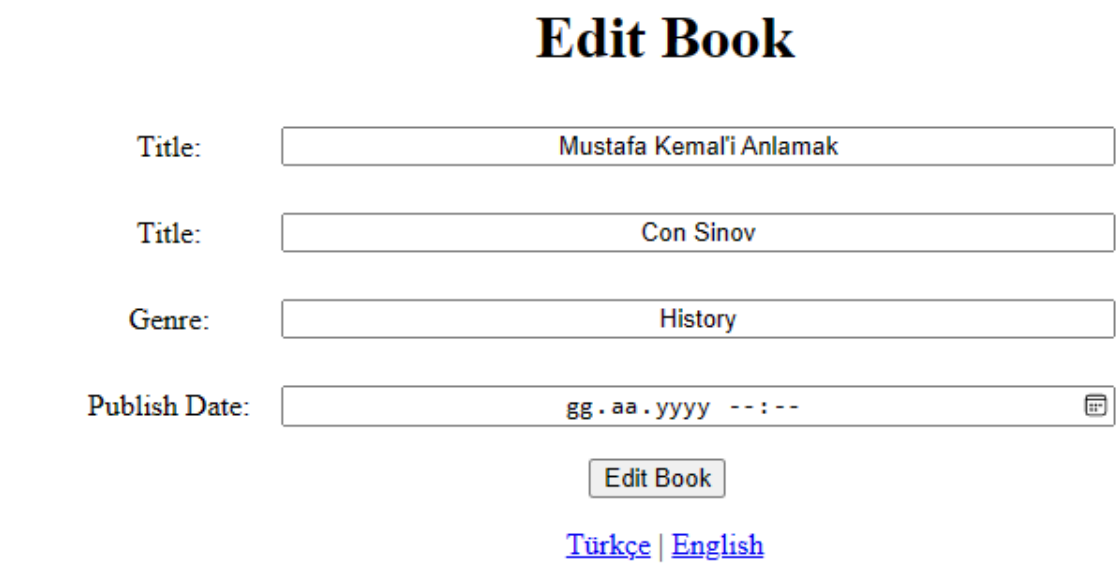
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.7.12</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>library</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>library</name>
	<description>Library project for Spring Boot</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-thymeleaf</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-validation</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
```

```
		</dependency>
		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<optional>true</optional>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
				<configuration>
					<excludes>
						<exclude>
							<groupId>org.projectlombok</groupId>
							<artifactId>lombok</artifactId>
						</exclude>
					</excludes>
				</configuration>
			</plugin>
		</plugins>
	</build>

</project>
```

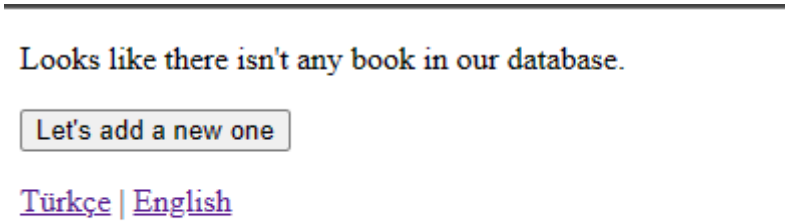
Add Book Screen:



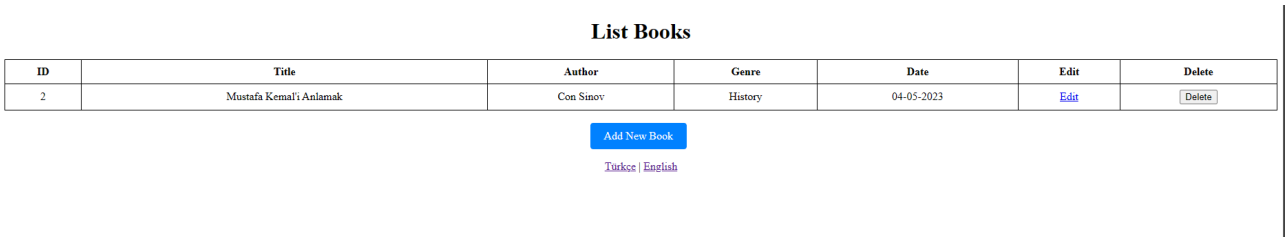
Edit Book Screen:



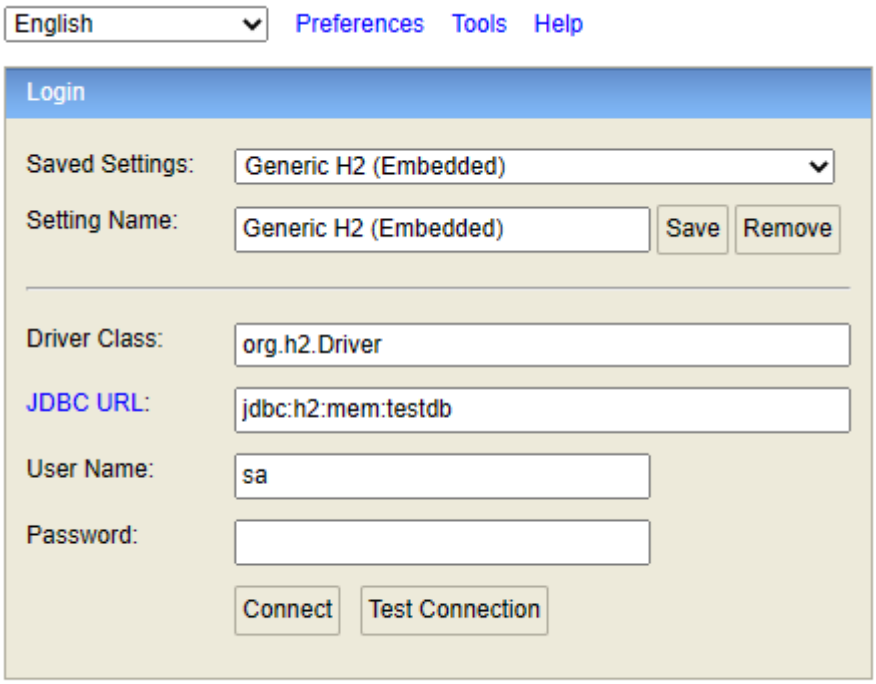
Empty Screen:



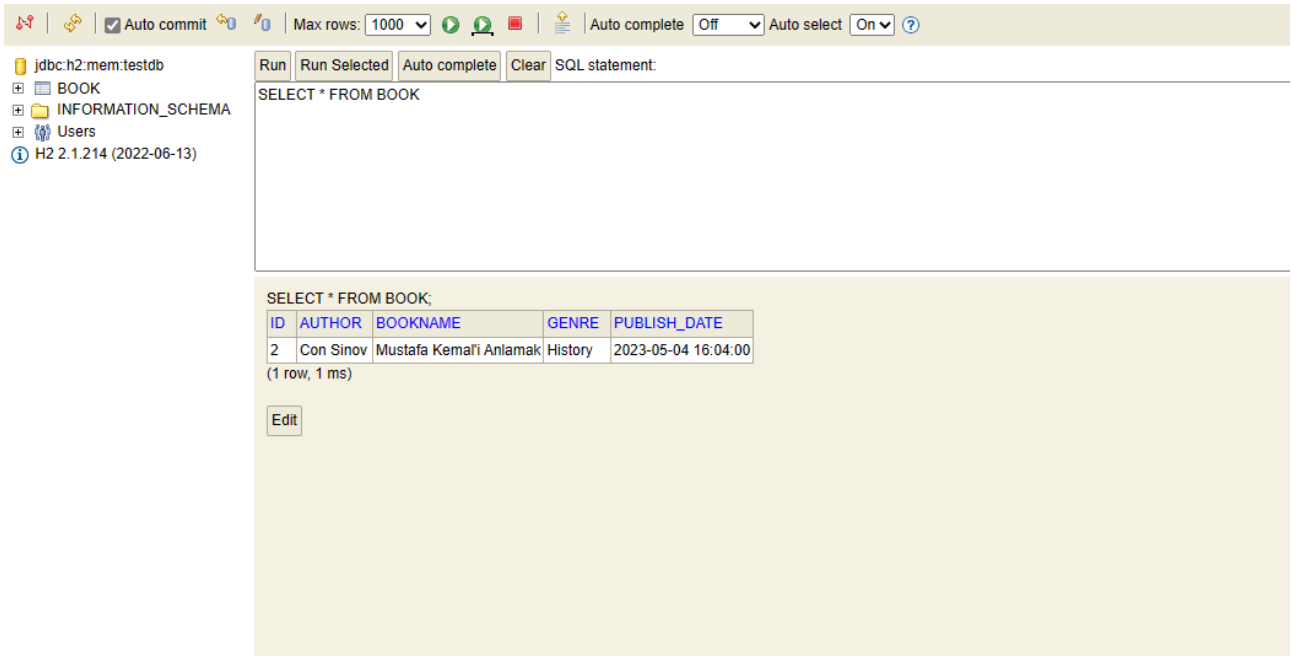
List Book Screen :



H2 database entry:



H2 Database:



Conclusion:Our application keeps the books in its database. Thus, it is possible to quickly view general information about the books saved in the database. Thus, we think that we have laid the foundations of an automated library management system. The aspects of our application that need improvement are that an application with more detailed book information and a more advanced interface can be designed. Search filters can be added and a more advanced database can be used.