**TABLE OF CONTENTS**

1. **Introduction………………….…………………………………………………………….3**
2. **Requirement ……………….…………………………..………………………………….4**
3. **Methodology……………….…………………………………………………………….5**
4. **Architecture Used: ………….………………………..………………………………..15**
5. **Conclusion ……………….……………………………….……………………………16**

1. **Introduction**

Graduation projects represent research, and practical application of knowledge, serving as a testament to the student's competence in their chosen field. However, the process of assessing these projects, usually undertaken by faculty members or appointed assessors, is not without its challenges. Traditional evaluation methods are increasingly disadvantaged due to their time-consuming nature, costly requirements, and limited potential for providing feedback.

In response to these challenges, the Graduation Project Evaluation Application (GPEA) was developed as a comprehensive solution to facilitate project submission, project submission time assignment, assessment and feedback processes. This web-based platform aims to address the inherent limitations of traditional assessment methods by randomly assigning presentation times based on professors' availability, thereby eliminating presentation time confusion and promoting transparency in the assessment phase. The application requires users to enter school and student information on their first login, which is then cross-checked with the school's student information system to verify user access, ensuring that only authorized individuals can use the platform's features.

Furthermore, GPEA extends its functionality to a mobile application designed specifically for professors, developed using modern technologies such as Flutter and Dart, and using a cloud-based database such as SQL Server Management Studio. Through the mobile app, professors can easily assess project progress and access all features available on the web platform. This integration of web and mobile interfaces ensures accessibility and usability for all stakeholders involved in the graduation project process.

Furthermore, GPEA extends its functionality to a mobile application specifically designed for professors, developed using modern technologies such as Flutter and Dart, and utilizing a cloud-based database like SQL Server Management Studio. Through the mobile app, professors can easily assess project progress and access all features available on the web platform. This integration of web and mobile interfaces ensures accessibility and usability for all stakeholders involved in the graduation project process.

The main purpose of this document is to introduce the GPEA platform, describe its key features and functionalities, and outline the benefits it offers to stakeholders within the academic community. By leveraging the power of technology and innovative design principles, GPEA aims to revolutionize the graduation project assessment paradigm and promote greater efficiency, fairness, and accountability in the assessment process.

1. **USED TECHNOLOGIES**  
   Through comprehensive research, the most suitable technologies have been identified to ensure the functionality of successful web and mobile applications in this project. Modern technologies such as Flutter, Dart, SQL Server Management Studio, Swagger, .NET, and Postman are employed.

**Mobile Application:**  
Flutter and Dart, developed by Google, enable the creation of high-performance, visually rich applications for both iOS and Android platforms using a single codebase through an open-source UI SDK. Dart, the programming language used in Flutter, is easy to learn, stable, and high-performing. This combination allows for rapid and efficient development of mobile applications, ensuring a seamless and visually appealing user experience.

**SQL Server Management Studio (SSMS):**  
SQL Server Management Studio is used for managing SQL Server databases and performing various database-related tasks. It provides a graphical interface for database management, query execution, and system maintenance, ensuring the efficient handling of database operations.

**Web Application:**  
.NET is a framework developed by Microsoft that is used for building modern, cloud-based, internet-connected applications. It provides a robust environment for developing high-performance web applications, ensuring scalability, security, and performance for the backend development.

**Swagger:**  
Swagger is used for API documentation and testing. It allows for the easy generation of API documentation, enabling developers to understand and interact with the API endpoints efficiently. Swagger helps streamline the development and testing of APIs, ensuring that the communication between different parts of the system is seamless.

The integration of these technologies enables the creation of dynamic, efficient, and scalable applications for both web and mobile platforms. By utilizing Flutter and Dart for mobile development, .NET for web development, SQL Server Management Studio for database management, and Swagger for API documentation, this project aims to deliver high-quality, user-friendly interfaces and robust data management capabilities. This combination of technologies facilitates the development of modern applications that meet current development standards and user expectations.

1. **Methodology**

**Authorization Processes**

The authorization mechanism in GPES utilizes **SQL Server Management Studio (SSMS)** to separate user roles and ensure secure access control.

**Role-Based Segmentation**

After logging in, users are assigned roles as either "Student" or "Professor." The **UserName** attribute corresponds to the student's ID number and is used to track user activities and manage access permissions. These details are managed via the relevant tables in the SQL Server database.

**SQL Server Integration**

To support role-based access control, the necessary database tables and relationships are configured in SQL Server. The database schema organizes data under the **UserLogin** table, which includes **Student** and **Professor** sections.

* **UserLogin Table:** Stores user credentials and roles.
* **Student and Professor Tables:** Contain specific information for students and professors and define access boundaries based on roles.

**Authorization Controls**

Authorization processes are carried out by a codebase integrated with SQL Server. This system verifies user login credentials and grants access to resources based on their roles.

* **Stored Procedures:** Used to query user data and implement access control.
* **Trigger Mechanisms:** Trigger-based checks ensure database-level monitoring for specific access scenarios.

This process guarantees that users can only access authorized resources based on their roles. The tools provided by SQL Server Management Studio enhance the security and manageability of these controls.

**Database and Table Creation**

The GPES database infrastructure is built using **SQL Server Management Studio (SSMS)**.

**Database Selection**

SQL Server was chosen for its robust data management capabilities, scalability, and performance optimization. Additionally, its security features, user access control, and relational data model align well with GPES requirements.

**Table Creation**

The required tables for structuring and managing data relationships were created in SQL Server:

* **AvailableDateTimeSlot Table:** Stores availability details for professors and projects.
* **BookedDates Table:** Maintains reserved presentation dates and related information.
* **Projects Table:** Contains details about graduation projects, such as titles, content, and student information.
* **Professors Table:** Includes information about professors, their expertise, and schedules.

**Relationships Between Tables**

Table relationships in SQL Server are defined using **Primary Key (PK)** and **Foreign Key (FK)** constraints:

* **Projects and Professors Relationship:** A "Many-to-Many" relationship allows multiple professors to be assigned to a single project.
* **AvailableDateTimeSlot and BookedDates Relationship:** Ensures seamless tracking of reservations based on available date and time information.

This structure enables GPES to effectively manage presentation dates and reservations. The user-friendly interface of SQL Server Management Studio ensures database administration is both secure and efficient.

**Login Operations**

A user-friendly login interface has been developed to facilitate secure access to the GPES platform. When accessing the login page, users are presented with a form to enter their credentials. The login process includes the following steps:

* **User Information Input:** Users enter their username and password in the designated fields and click the "Login" button.
* **Automated Login with Selenium:** Selenium interacts with the remote login portal at [https://cats.iku.edu.tr/portal](https://cats.iku.edu.tr/portal" \t "_new) on the web platform. The login form is automatically filled with the credentials provided by the user, and the login process is completed.
* **Verification Process:** After the credentials are submitted, GPES initiates an internal verification process. This process ensures the authenticity of the user's credentials without exposing direct database queries or transactions.
* **API Usage via Mobile Platform:** The mobile platform sends a request to the web through the API and performs the login process on the back-end.
* **Authentication API:** An API that centralizes the authentication logic simplifies the process for developers and ensures consistency across different platforms. Ngrok is used to open the API to external platforms, facilitating seamless integration.

**Matching Algorithm**

**Main Method:** ScheduleTeamsPresentationsOptimizedBacktracking

This method orchestrates the scheduling process and performs the following steps:

1. **Data Preparation:**
   * Fetches all teams and professor availability data.
   * Converts teams to a list and prepares separate lists to track scheduled presentations and unscheduled teams.
2. **Invoking the Backtracking Algorithm:**
   * Calls the AssignPresentationsBacktracking function to try and find an optimal solution.
   * **If a solution is found:**
     + The presentation schedule is saved to the database.
     + Results are returned to the user.
   * **If no solution is found:**
     + The team that couldn’t be assigned is removed from the list, and the algorithm retries with the remaining teams.
3. **Error Handling:**
   * If an error occurs during the process, a 500 status code with an error message is returned to the user.

**Backtracking Function: AssignPresentationsBacktracking**

This function implements the backtracking algorithm to assign teams to presentation slots:

1. **Base Case:**
   * If there are no more teams to assign:
     + Updates the best schedule if the current schedule has more presentations than the previous best.
     + Returns true, indicating success.
2. **Selecting Available Time Slots:**
   * Retrieves the availability of the current team's advisor.
   * Sorts the advisor's available slots by date and start time.
3. **Checking Professor Availability and Generating Combinations:**
   * For each available time slot, identifies at least two other available professors.
   * Generates all possible combinations of two professors from the available professors.
4. **Attempting a Slot Assignment:**
   * Assigns the team to the selected time slot with the advisor and two additional professors.
   * Recursively calls the function for the remaining teams.
5. Backtracking:
   * If no solution is found:
     + Reverts changes (removes the assignment and restores the availability).
     + Continues to the next combination or time slot.

**Combination Function: GetCombinations**

This helper function generates all possible combinations of a specified size from a given array. It is used to create all pairs of professors who can attend the presentation along with the advisor.

**Console Outputs**

The code uses multiple Console.WriteLine statements throughout to log:

* The progress of the algorithm,
* Teams and combinations being tried,
* Successful or failed attempts,
* Updates to the maximum scheduled count.

These logs are intended to debug and monitor the scheduling process in detail.

**Functionality: SyncUsersWithProfessors**

**Purpose:**

The endpoint ensures that the relationship between users and professors is synchronized. If a user-professor mapping is missing in the database, it adds the missing relationship.

Steps:

1. **Fetch Users with Professors:**
   * The method retrieves a list of users and their corresponding professors using the \_userAppService.GetUsersWithProfessorsAppAsync() method.
2. **Check and Add Relationships:**
   * For each user-professor pair:
     + It checks if the relationship already exists using \_professorsUsersAppService.ProfessorsUsersExistsAppAsync().
     + If the relationship does not exist, it creates a new record with \_professorsUsersAppService.AddProfessorsUsersAppAsync().
3. **Return a Success Response:**
   * After processing all records, a success message is returned.
4. **Error Handling:**
   * If an exception occurs during processing, a 500 status code is returned along with the error message.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**A screenshot of a computer

Description automatically generated** **A screenshot of a computer

Description automatically generated**

**A computer screen shot of a computer

Description automatically generated**

1. **Architecture Used:**

**Onion Architecture**

Onion Architecture was selected for the GPES system due to its clean separation of concerns, maintainability, and scalability. The architecture is designed to address common challenges in large applications, such as tight coupling, difficulty in testing, and managing complex business logic.

**Why Onion Architecture?**

1. **Separation of Concerns:**Onion Architecture promotes a clear division between different layers of the application, making it easier to isolate and manage dependencies. The core business logic is placed at the center of the architecture, with the outer layers depending on it, rather than the other way around. This allows for better organization of code, reducing complexity.
2. **Testability:**By decoupling the business logic from the external infrastructure, Onion Architecture makes it easier to write unit tests. The core logic remains isolated from the implementation details of data access, user interfaces, and other external components. This makes the system more testable and ensures higher code quality.
3. **Scalability and Flexibility:**The Onion Architecture allows the system to scale and adapt to changing requirements over time. Since the dependencies are handled through interfaces and dependency injection, new functionality, components, or external services can be integrated without altering the core business logic. This makes the system highly flexible for future growth.
4. **Maintainability:**As the core logic is separated from the infrastructure and user interface layers, making changes to the business logic does not directly affect other parts of the system. This results in easier maintenance and faster implementation of new features or modifications to the application.
5. **Improved Dependency Management:**Dependencies are handled in a way that inner layers do not depend on outer layers. The outer layers, such as the UI and data access, interact with the inner layers through interfaces, ensuring that the core logic remains independent and reusable.
6. **Conclusion**

In conclusion, the development of the Graduation Project Evaluation System (GPES) represents a significant leap forward in modernizing educational assessment practices. By integrating cutting-edge technologies such as Flutter, Dart, SQL Server Management Studio, Swagger, and .NET, this comprehensive platform aims to revolutionize the evaluation process for academic projects.

Through the utilization of Flutter and Dart, the mobile application aspect of GPES offers a high-performance, cross-platform solution with a single codebase. This not only enhances development efficiency but also ensures a seamless user experience for both educators and students. SQL Server Management Studio is used for efficient and secure database management, ensuring smooth data storage, retrieval, and management.

On the web front, the adoption of Onion Architecture provides a robust and scalable foundation for the GPES platform. This architecture ensures modularity, testability, and maintainability of the web application codebase, facilitating efficient project submission, evaluation, and feedback mechanisms. The Onion Architecture allows for clear separation of concerns, promoting better system organization and easier maintenance.

The integration of Swagger enhances the system by providing detailed API documentation, enabling easier integration with external systems and services. It facilitates seamless communication between the web and mobile platforms and ensures that developers have a clear understanding of how to interact with the APIs.

The technical specifications and functionalities demonstrated in both the mobile and web components underscore the versatility and effectiveness of the chosen technologies. From user authentication to project submission and evaluation, each component is meticulously designed to optimize user experience and streamline the evaluation process.

While certain limitations and constraints exist, such as specific evaluation criteria and validation requirements, GPES strives to maintain data integrity and accuracy in evaluation outcomes. Furthermore, future enhancements and refinements can be made to optimize system performance and user experience, ensuring seamless integration into academic workflows.

In summary, GPES represents a comprehensive platform that not only enhances the assessment process for academic projects but also promotes transparency, efficiency, and accountability. By leveraging technology and innovative design concepts, GPES holds significant potential to revolutionize the graduation project assessment landscape, ultimately contributing to the advancement of academic excellence and innovation.