ToGather

Design Document

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

**Design structure**

The architectural design of the ToGather App is composed of three layers which are UI Layer, Domain Layer and Technical Services Layer in accordance with the separation of principles and cohesiveness. In the UI Layer, React Native and React frameworks are included. In Domain Layer, application logic representing use cases and functionalities of the application is included such as User, Task, Profile, Calendar, Automation Engine and Recommendation Engine. Finally, as third Layer, Technical Services that aims to serve to other layers is included. In this layer, common services such as Logging, Django Security, Django, Backup and Recovery, MongoDB, RESTful API, Django Authentication, MySQL, NoSQL, Caching, Google oAuth, AWS EC2, NGINX, i18n Library and moment.js Library will be used. The detailed diagram is given in Figure 1.

ekran görüntüsü, metin, dikdörtgen, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1: Layered architecture

**Subsystems**

There are no subsystems within the system.

**Patterns**

**[Authentication and Authorization]**

**Overview**

This pattern will be used in all log-in and user profile management process in the application. The intent is to provide secure authentication and authorization mechanism which aims to verify user identity, enforce role-based access control and manage account. In this way CIA triangle (Confidentiality, Integrity and Availability) will be satisfied. Motivation is to maintain reliability while using the application. By providing secure AAA (Authentication, Authorization and Accounting) mechanisms, the sensitive information will be ensured to be protected and unauthorized access from suspicious users will be prevented. Since the commonly used standards will be used, this pattern is universally applicable to similar applications and essential for protecting sensitive data.

**Structure**

**Users:** User data with sensitive information such as username, password and personal information. Users are responsible for initiating the authentication requests and managing their profiles.

**Authentication and Authorization Service:** The data protected will be user credentials, authentication tokens, authorization tokens, user permissions and roles. These services will be responsible for validating user credentials, providing authentication tokens, session management, security policy enforcement, managing the authorization processes, managing user Access according to their defined roles.

**Third Party Applications:** The data protected will be OAuth tokens. Google OAuth services will be the third-party services supporting this pattern in the application. These services will responsible for interacting with the authorization service to gain Access tokens. Also, they are supposed to request Access to user data via OAuth protocols.

**Behavior**

**User Authentication:** User Authentication process consists of User Initiation, Validation of Credentials and Token Acceptance. In user initiation process, User and Authentication Service play an important role. In credential validation, user and authentication service play role. Finally, in token acceptance, authentication service and user play an important role.

**Authorization and Access Control:** This scenario consists of role-based Access control (RBAC) and token-based authorization when used with third party applications. In role-based authorization, user and authorization service play role. In token-based authorization, user and authorization service play an important role.

**Example**

In every login process necessary for actualizing the use-case scenarios, authentication and authorization pattern will be used.

**Requirement realizations**

In this section, firstly a class diagram in Figure 2 is introduced to show the relationship of classes for realized use cases. Since we are developing our application by using Django framework, all classes are inherited from django.db.models “Model” class.

Secondly, sequence diagrams for each realized use case are introduced.

metin, diyagram, plan, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2: Class Diagram

**[Use Case Realization 1 - Create a Task]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User selects the current Schedule. User presses the corresponding button to initiate task adding process. User fills the required fields then finishes task addition process.

Relationship: Initiates the task creation process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to create a task.
2. System displays the schedule.
3. User selects manual task creation option.
4. System displays task creation form.
5. User fills the required fields for manual task creation in the form, explained in the Glossary.
6. System creates task.
7. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 3.

metin, ekran görüntüsü, paralel, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3: Create a Task Sequence Diagram

**[Use Case Realization 2 - Update Task]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User selects the current Schedule. User searches for the task to be updated from the Schedule view then selects update. From the menu displayed, user fills in the required fields then finishes the process.

presses the corresponding button to initiate task adding process. User fills the required fields then finishes task addition process.

Relationship: Initiates the task update process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to update a task.
2. System displays the schedule.
3. User selects a task to modify.
4. System displays task detail in an editable form.
5. User makes some changes.
6. System updates the task.
7. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 4.

metin, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4: Update Task Sequence Diagram

**[Use Case Realization 3 - Remove Task]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User selects the current Schedule. User searches for the task to be removed from the Schedule view then selects remove option. From the schedule displayed, user checks the removal then finishes the process.

Relationship: Initiates the task removal process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to remove a task.
2. System displays the schedule.
3. User selects a task to remove.
4. System asks for confirmation.
5. User confirms removal.
6. System removes the task.
7. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 5.

metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5: Remove Task Sequence Diagram

**[Use Case Realization 4 - Set User Profile]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User selects the profile. User views the personal information to be updated. User fills in the fields to be updated then finishes the process.

Relationship: Initiates the set profile process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to set user profile.
2. System displays user’s profile.
3. User selects update option.
4. System displays user profile creation form.
5. User fills the personal information fields.
6. System updates user profile.
7. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 6.

metin, ekran görüntüsü, diyagram, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6: Set User Profile Sequence Diagram

**[Use Case Realization 5 – Handle Friendship Request]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User views requests. User accepts the friend request.

Relationship: Initiates handling friendship request. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to handle other users’ friendship requests.
2. System displays existing activities.
3. User accepts the desired user’s friendship request.
4. System indicates accepting request.
5. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 7.

metin, diyagram, paralel, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7: Handle Friendship Request Sequence Diagram

**[Use Case Realization 6 – Review Existing Task]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User reviews an existing task.

Relationship: Initiates reviewing existing task process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to review an existing task.
2. System displays the task list.
3. User selects task to review.
4. System displays detailed information about the selected task.
5. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 8.

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 8: Review Existing Task Sequence Diagram

**[Use Case Realization 7 – Add Friend]**

**View of participants**

**Users**

The participants consist of users who are professionals with tight schedules and active social life and university or higher degree students.

Behavior: User views friend list. User adds friend.

Relationship: Initiates adding friend process. Interacts with the user interface.

Attribute: username, first name, last name, email, password

**Basic scenario**

The flow belonging to the main success scenario of the use case is listed below.

1. User wants to add other users as friend.
2. System displays existing users.
3. User browses the desired user.
4. System displays search results of user information.
5. User selects the desired user.
6. System displays the other user’s profile.
7. User sends invitation to add as a friend.
8. System indicates invitation is sent.
9. The use case ends successfully.

Sequence diagram for this use case’s main success scenario is given in Figure 9.

metin, diyagram, paralel, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 9: Add Friend Sequence Diagram

# Revision Table

|  |  |  |
| --- | --- | --- |
| **Revision** | **Description** | **Date** |
| 1.0 | First revision | 24/11/2023 |
| 1.1 | In Requirement Realizations section, Class Diagram and sequence diagrams for each realized use case are added, according to advisor’s Iteration 2 Work Products Feedback. | 15/12/2023 |
| 1.2 | Changes applied according to ToGather document. | 22/12/2023 |