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 servet o 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**

**[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. The application user interface is displayed to the User.
2. User selects the schedule.
3. User presses “+” button to add new task additional to the existing tasks.
4. System displays two options: “Manual Task (Marked as Default)” and “Automated Task”
5. User clicks “Next” button.
6. System displays a menu including the necessary fields to be filled.
7. User fills the required fields for manual task creation, explained in the Glossary.
8. User presses Finish button.
9. System displays the schedule.
10. The use case ends successfully.

**[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. The application user interface is displayed to the User.
2. User selects the schedule.
3. User find an existed task to update options.
4. System displays the task with detailed form.
5. User clicks “Next” button.
6. System displays a menu including the necessary fields to be filled.
7. User updates the fields by their choices.
8. User presses Update button.
9. System displays the schedule.
10. The use case ends successfully.

**[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. The application user interface is displayed to the User.
2. User selects the schedule.
3. User find a desired existed task to remove it.
4. User clicks “delete” button.
5. System removes the requested task.
6. System displays the schedule without removed task.
7. The use case ends successfully.

**[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. System displays application user interface to the User.
2. User selects the Profile Tab from Menu.
3. System displays Profile page.
4. User presses Update button to set the personal information.
5. System displays a menu including the personal information fields.
6. User fills the personal information fields.
7. Finally, User presses Finish button to save new information.
8. System displays updated Profile page.
9. The use case ends successfully.

# Revision Table

|  |  |  |
| --- | --- | --- |
| **Revision** | **Description** | **Date** |
| 1.0 | First revision | 24/11/2023 |