ToGather

Architecture Notebook

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**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

In Section 2, architectural goals and philosophy are mentioned. In Section 3, assumptions about user environment and general environment are explained and other dependencies are mentioned. In Section 4, requirements that effect the core architecture are introduced. In Section 5, architectural design decisions, their justifications and related requirement IDs are mentioned. In Section 6, architectural mechanisms are explained. In Section 7, domain model is introduced as key abstractions. In Section 8, layered framework is introduced and lastly, in Section 9, architectural views are introduced with diagrams.

# Architectural goals and philosophy

* Deployment: Within the scope of this project, only the server-side deployment will be considered, and client will be a compatible web browser. Therefore, the deployment environment will not be diverse. Our application will be developed by adopting a microservice architecture.
* Adaptation: With microservice architecture, operating system dependency will be minimized. In that way, the system will be adaptable to different environments. Also, by developing a web-based application, client-side computations will be kept to a minimum and any device with a simple browser will be able to run the application.
* Performance issues: CDN (Content Delivery Network) will be used to provide seamless accessibility, so the users from different geographic locations will access the application regardless of the distance to the server. The application shall be capable of scaling under varying user loads.
* Maintenance: Long-term maintenance for infrastructure will be provided by the cloud provider. For application long-term maintenance, since layered architecture will be used, it will be an advantage for troubleshooting in isolated parts of the architecture.
* Hardware dependencies: For mobile application, performance of the system will be affected by the resource limitation of users’ devices. Performance of the system will be affected by server’s resource limitation for both the web and mobile application.

# Assumptions and dependencies

* It is assumed that users are professionals or university or higher degree students.
* It is assumed that users are using compatible browser to reach the web application, such as Chrome, Safari, Opera, Mozilla Firefox, Edge, and their stable versions.
* For mobile application, it is assumed that users are using iOS or Android operating systems, and their stable versions.
* Mobile application is dependent on iOS and Android operating systems.
* For web application, architecture is dependent on web-based design decisions.
* It is assumed that Cloud service provider will ensure reliability.
* It is assumed that user devices’ resources are sufficient, for mobile application performance.
* It is assumed that servers’ resources are sufficient, for web and mobile application performance.
* It is assumed that users are experienced in using other scheduling applications before, such as Outlook Calendar, Google Calendar. It will ensure better user experience for users.
* It is assumed that users will regularly update their calendar, so that it will represent the users’ actual schedule.
* The effectiveness of automated scheduling depends on users keeping their calendar up to date.
* It is assumed that users will have a stable internet connection for real-time synchronization, updates and getting instant notifications for both the web and mobile application.
* User privacy and data protection mechanism will be compliant to KVKK. Database design will be dependent on this regulation.
* The application will offer seamless integration with other calendar applications (Outlook Calendar, Google Calendar) for synchronization. Synchronization efficiency will be dependent on the capabilities they offer.
* It is assumed that the event organizers keep their data accessible and up to date.
* Deployment architecture will be dependent on the server’s hardware specifications.
* Application is dependent on the chosen development frameworks and libraries in terms of receiving regular security updates and maintaining compatibility.

# Architecturally significant requirements

* TG-Req-001] The system must provide secure authentication mechanisms to verify user identity and enforce role-based access control.
* [TG-Req-004] The system must maintain comprehensive logs of all user activities, system events, and security-related incidents. These logs must be securely stored and readily accessible for auditing and analysis purposes.
* [TG-Req-005] To ensure data integrity in the system failures, the system must implement regular data backup procedures, providing safety against data loss.
* [TG-Req-006] The system should integrate with third-party services, such as event organizations, social media platforms, and external data sources, to expand the system's functionality.
* [TG-Req-007] The system must be designed to scale for ensuring optimal performance when user loads and data volumes grow.
* [TG-Req-008] The system must provide efficient search and query capabilities, enabling users to locate and retrieve specific information from the application.
* [TG-Req-017] The system's user interfaces must be responsive and mobile-friendly, ensuring availability across various devices and screen sizes.
* [TG-Req-018] The system should be accessible on multiple platforms, including iOS, Android, and web.
* [TG-Req-023] The system must support Turkish and English languages.
* [TG-Req-024] The system must support local date formats, regional settings and time zones.
* [TG-Req-027] The UI shall comply with WCAG 2.1 accessibility standards to ensure the usability by individuals having sensitivity and disabilities.
* [TG-Req-038] The UI shall be compatible with commonly used web browsers such as Chrome, Safari, Opera, Mozilla Firefox, Edge, and their stable versions, and iOS and Android platforms at the time of this document.
* [TG-Req-041] The availability target of the system shall be 99.9% uptime at minimum.
* [TG-Req-045] Load balancing shall be implemented to distribute the traffic across multiple servers or replicates of services according to the capacity at that time.
* [TG-Req-053] The customer data backups shall be taken to a separate storage location daily.
* [TG-Req-056] Cloud services shall be used to provide geographical redundancy of the system.
* [TG-Req-061] The web application's web pages shall load within 2-3 seconds for at least 90% of users under typical user loads.
* [TG-Req-062] The mobile application's pages shall load within 1-2 seconds for at least 90% of users under typical user loads.
* [TG-Req-063] The application shall support minimum of 1000 concurrent users.
* [TG-Req-064] The application shall handle a minimum of 1000 transactions per second.
* [TG-Req-065] Search queries shall return results in at most 500 milliseconds for more frequent searches.
* [TG-Req-066] Database queries shall execute in at most 2 seconds for complex queries.
* [TG-Req-068] For repeated database queries, caching mechanisms shall achieve a hit rate of 95 %.
* [TG-Req-070] The users from different geographic locations shall access the application regardless of the distance to the server. CDN (Content Delivery Network) shall be used to provide seamless accessibility.
* [TG-Req-072] The application shall be capable of scaling under varying user loads.
* [TG-Req-073] The application shall be designed and built in a modular perspective to provide isolation between components.
* [TG-Req-076] To identify and resolve issues, the system shall log every incident, errors, exceptions with detailed information, subject, action, and timestamp.
* [TG-Req-085] User interface must be responsive which means it should be adaptive to different screen sizes from mobile to desktop.
* [TG-Req-103] To provide a single-click and secure sign-up and log-in, third-party authentication services (such as Google OAuth) should be used.
* [TG-Req-107] To minimize server-side hardware requirements, cloud services should be used.
* [TG-Req-108] Client-side computations should be kept to a minimum to ensure that any device with a simple browser is able to run the application.
* [TG-Req-109] AWS, Django and React utilities will be responsible for providing basic communication interfaces to our system.
* [TG-Req-124] Open-source software implementation languages and frameworks will be used in this project. Based on the time of the preparation of this document, the stable versions will be used.
* [TG-Req-125] As backend, software implementation language will be Python.
* [TG-Req-126] As a backend framework Django will be used.
* [TG-Req-127] As frontend, software implementation language will be Javascript (HTML and CSS will be used too).
* [TG-Req-128] As frontend framework, React will be used.
* [TG-Req-129] As a relational database management program, MySQL will be used.
* [TG-Req-130] As NoSQL database management program, MongoDB will be used.
* [TG-Req-131] For development, testing and production environment, cloud services (AWS, GCP etc.) will be used. For development purposes, PaaS or IaaS services will be received according to use cases.
* [TG-Req-132] Users who want to use ToGather Application, must sign up with an agreement to KVKK (Kişisel Verilerin Korunumu Kanunu) and Accepting Cookies to enhance site navigation, analyze site usage, and assist in our marketing efforts.
* [TG-Req-140] Web Content Accessibility Guidelines (WCAG) 2.1

“Web Content Accessibility Guidelines (WCAG) 2.1 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content more accessible to a wider range of people with disabilities, including accommodations for blindness and low vision, deafness and hearing loss, limited movement, speech disabilities, photosensitivity, and combinations of these, and some accommodation for learning disabilities and cognitive limitations; but will not address every user need for people with these disabilities. These guidelines address accessibility of web content on desktops, laptops, tablets, and mobile devices. Following these guidelines will also often make Web content more usable to users in general.”

# Decisions, constraints, and justifications

|  |  |  |
| --- | --- | --- |
| **Design Decision** | **Justification** | **Requirement ID** |
| Using Django Auth and Google Auth | For secure authentication, Django Auth will be used because of its compatibility with Django and Google Auth will be used because it is a common third party authentication service. | TG-Req-001  TG-Req-103 |
| Using logging module | Logging module will be used. All the important logs will be written to a database. | TG-Req-004  TG-Req-076 |
| Getting copies of database on regular basis and storing in different geographic locations | Database will be copied on a regular basis and stored in different vendors in different locations. It will prevent failure due to storing in only one location and the failure of that location. | TG-Req-005  TG-Req-053 |
| Adding module that will collect data through APIs | A module will be added, which will collect data through APIs of third-party websites. If there are no APIs for the websites, the module will write web crawlers specific to the websites. | TG-Req-006 |
| Load balancing | Load balancing and parallelism will be applied. Since Cloud is used, it will be very easy. | TG-Req-007  TG-Req-072 |
| Using NoSQL for database | NoSQL will be used for database, because it has faster read and write features. It is more suitable for user search. | TG-Req-008  TG-Req-066 |
| Using React for web application, React Native for mobile application | For web application, React will be used to make UI, because it is very easy to use and a modern framework. It is a good choice for responsiveness and compatibility, since responsive design principles will be used.  For mobile application, React Native will be used to make mobile native UIs. | TG-Req-017  TG-Req-038  TG-Req-085  TG-Req-128 |
| Using RESTful APIs | In addition to using React and React Native explained for TG-Req-017, RESTful APIs will be used. By using RESTful APIs, modularity of backend will be ensured. Also, it will make backend compatible for many platforms. Because the format that RESTful APIs turn into and operating logic is very broad and easily adaptable. | TG-Req-018 |
| Using i18n library | i18n library will be used. This library will be integrated with React. With the help of integration of i18n and Google Translate, automatic translations will be made. | TG-Req-023 |
| Using moment.js library | moment.js library will be used. This library automatically detects user’s time zone and sync it to other time zones or server’s time zone. | TG-Req-024 |
| Using proper HTML elements | Proper HTML elements will be used. For page styling, a proven css framework, such as bootstrap, will be used. | TG-Req-027  TG-Req-140 |
| Using Cloud for system capacity | In different geographical locations, different cloud instances will be created. The system will always be updated according to the existing user load and trend. | TG-Req-041  TG-Req-056  TG-Req-070 |
| Using Cloud and NGINX together | Cloud will be used to be able to add another server due to increasing traffic. At the same time, NGINX will be used to ease load balancing. | TG-Req-045 |
| Using NGINX, NoSQL and React | NGINX will be used to ease load balancing. NoSQL will be used since it is fast. React will be used for its virtual dom feature, which will refresh only the changed data when the whole page is refreshed, instead of refreshing all the data including unchanged ones.  These three will be used at the same time to achieve required page loading durations under certain user loads. | TG-Req-061  TG-Req-062 |
| Choosing CPU and RAM amount | Cloud resources’ (CPU and RAM) amount will be chosen according to the requirement. | TG-Req-063  TG-Req-064 |
| Using caching mechanisms | Caching mechanisms will be used. If caching is applied in proxy server for most frequently sent queries, then it won’t be necessary to send each query to web server every time. Caching will be done at client side too. | TG-Req-065  TG-Req-068 |
| Dividing processes into logical parts | Each process will be divided into logical parts. To manage this in backend, separate Django apps will be written for each logical part. | TG-Req-073 |
| Using AWS EC2 | AWS EC2 will be used, because it is free and very easy to adapt to React and Django. | TG-Req-107  TG-Req-131 |
| Using servers which are dedicated to handle processes that take long time and require lot of resources | Processes that take long time and require lot of resources such as optimization process will be handled at server side by servers which are dedicated for these kinds of processes. | TG-Req-108 |
| Using Django, React, AWS | Django and React will be used since they are open-source, modern and comprehensive. Django’s another advantage is its predefined security functionalities.  AWS will be used since it is common throughout the world and gives free tier service. | TG-Req-109 |
| Using Python, Javascript, Django, React and MongoDB | Python will be used, since the development team have a good knowledge of Python language and it has a modern and very comprehensive framework just like Django. Javascript, Django, React and MongoDB will also be used since they are open source. | TG-Req-124  TG-Req-125  TG-Req-126  TG-Req-127 |
| Using MySQL | Since MySQL is an open-source relational database management program, it will be used in this project. | TG-Req-129 |
| Using MongoDB | Since MongoDB is an open-source NoSQL database management program, it will be used in this project. | TG-Req-130 |
| Secure data storage  (Using Django) | Passwords will be stored in hashed format, which Django does automatically.  All data in database will be stored securely and won’t be shared to other users or third parties without user’s consent. | TG-Req-132 |

Table 1: Architectural design decisions and related requirement IDs

# Architectural Mechanisms

## Authentication and Authorization Mechanism:

Purpose: System will provide secure authentication and authorization mechanism which aims to verify user identity, enforce role-based access control and manage account.

Attributes: Authentication method (username and password, single sign-on etc.), account management (creating and recovering user account), role-based access control, session management, authentication security (API security), policy-based access control (accepting cookies like KVKK, etc.), logging, certification.

## Backup and Recovery Mechanism:

Purpose: The system will implement regular data backup procedures, providing safety against data loss to ensure data integrity in the system failures.

Attributes: Regular data back-up (regular basis and storage location), integrity check, logging, auditing, pulling backup (last version).

## Multi-Language Support Mechanism:

Purpose: The system will support English and Turkish languages to make application more accessible to global and local users.

Attributes: Localization (Turkish), Globalization (English), Dynamic Content Adjusment (alignment due to language), UI Translation, Sensitivity (Culturally), Easy Switching (Between Languages)

## Alerting Mechanism:

Purpose: The system will have Alerting Mechanism according to needs and performance to improve effectiveness and availability of users.

Attributes: Event Reminders (via Email), Customizable Reminder, Location-Sensitive Reminder (Based on geographic location), Priority (Reminder frequency), Snooze and Postpone, Update Notification (Add, delete people/location and change time)

## APIs Mechanism:

Purpose: The system will collect data from third party event organizers’ websites through APIs.

Attributes: Data exchange in multiple data formats, interoperability (compatibility with other platforms), secure data transformation between client and server, caching (for frequently requested data), logging

## Data Synchronization Mechanism:

Purpose: The system will ensure that data consistency is preserved across multiple devices and platforms.

Attributes: Support transactions, validate data, synchronization method (two-way, one-way etc.), logging, auditing, interoperability (for cross-platform data synchronization)

# Key abstractions

diyagram, plan, metin, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1: Domain Model

# Layers or architectural framework

metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2: Layer framework

# Architectural views

* **Logical view:**

metin, diyagram, plan, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3: Logical view

* **Deployment view:**

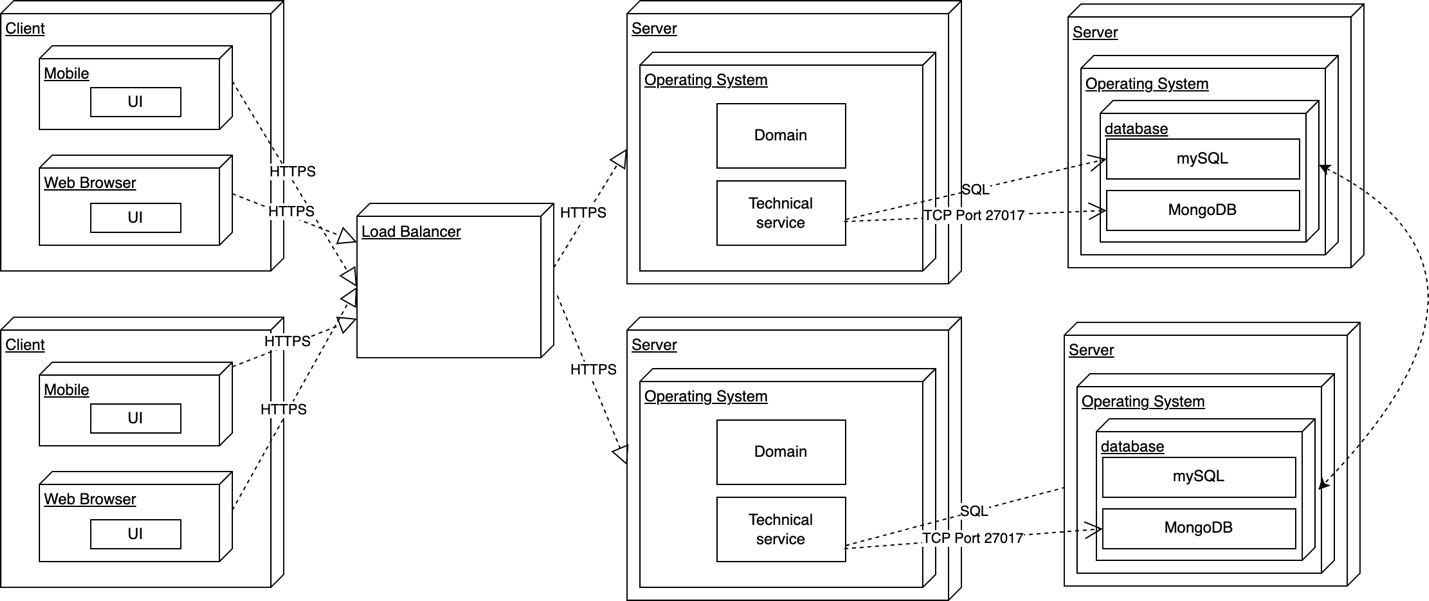


Figure 4: Deployment view

* **Use Case view:**



Figure 5: Use Case view

* **Data view:**

**diyagram, simetri, bakışım, çizgi, kalıp, desen, düzen içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Figure 6: Data view

Revision Table

|  |  |  |
| --- | --- | --- |
| **Revision** | **Description** | **Date** |
| 1.0 | First revision | 24/11/2023 |
| 1.1 | Changes are applied according to ToGather\_Review document. | 30/11/2023 |
| 1.2 | -Data View is added to Section 9, according to advisor’s Iteration 2 Work Products Feedback.  -Domain Model is updated in Section 7.  -Layer Framework is updated in Section 8.  -Logical View and Use Case View is updated in Section 9. | 13/12/2023 |