Bilkent University
Department of Computer Engineering

Senior Design Project

T2308

Perfent

Detailed Design Report

21901631, Beste Güney, beste.guney@ug.bilkent.edu.tr
21802838, Bora Çün, bora.cun@ug.bilkent.edu.tr
21903474, Cemal Faruk Güney, faruk.guney@ug.bilkent.edu.tr
21801831, Çağrı Eren, cagri.eren@ug.bilkent.edu.tr
21902461, Gamze Elif Çenesiz, elif.cenesiz@ug.bilkent.edu.tr
Supervisor: Cevdet Aykanat
Course Instructors: Erhan Dolak, Tağmaç Topal

12.03.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

Detailed Design Report

Perfent

1 Introduction

1.1 Purpose of the System

Perfent targets any group that wants to hang out and attend events together. It optimizes the process of finding a slot that is available for every member and also comes with event suggestions that might interest the group members. The event suggestions will combine the events from different web pages and different event providers which will also cause an improvement in the experience of browsing events. This way, the users do not have to visit several pages to find an event that fits their preferences. Perfent aims to bring incremental innovation to the process of organizing group events by optimizing the process of scheduling, organizing, and finding events that are a great fit for a group. It is aimed to enhance product performance by distinguishing features and functionality. Digital business optimization will be applied to create a better user experience and improve the productivity of the system.

The users will have their individual schedules on the system where they can indicate their busy and free time slots. For groups, these schedules will be combined to find a free slot for the whole group. In addition, based on the preferences of the group members, Perfent will recommend a proper event for the group. All group members will be notified about the free slots, the recommended event and specify whether they want to attend that event or not.

1.2 Design Goals

1.2.1 Maintainability

The application will have the necessary documentation and tools set up to enhance maintainability which is the ease of modifying a component or a system to correct faults and improve performance or other attributes [1]. To satisfy such needs our application will use the following metrics and target a maximum of 5% code duplication threshold, a minimum of 80% unit test coverage, and a maximum cyclomatic complexity of 20 for each unit [2].

1.2.2 Availability

The application will be available for most of the time of its lifetime. Our application will aim for a minimum availability of 99% during its lifetime. Most services on the internet fall between 99% and 100% of availability and our application targets to be like one of those services at the bare minimum [3].

1.2.3 Usability

The user interface of the application should be easy to manage, simple to use, and usable. It will ensure that all of the pages of the user interface can be understood at a reasonable level and traversed in a maximum of 1 minute.

1.2.4 Safety

Any private personal information entered into the system by the user such as interests or addresses will not be disclosed to the public and will be safeguarded by the servers.

Passwords entered into the system will be hashed with effective hashing algorithms that further protect them [4].

The application will have the necessary features to ensure that users are going to hazard-free events with hazard-free users.

1.2.5 Scalability

Our servers should be able to scale when it is necessary and handle the requests incoming from 5,000 concurrent users seamlessly and without any repercussions to the users using the website and the availability of any of Perfent's functionality. It should be able to load balance the coming traffic when the traffic gets heavy since not managed traffic can cause lags in the system and lag can be a determinant factor in losing a customer [5].

1.2.6 Performance

The application will satisfy the user's waiting time expectations and prevent users from bouncing off our website. The application will target a 2-second loading time threshold with a 6-7% bounce rate for the initial (entry) loading of the website [6]. Then, for each loading of the other pages, it will target the 1-second loading time with a 6-7% bounce rate [6]. Finally, for other actions of the user in the user interface that do not include server interactions, it will target the maximum action time of 100ms.

1.2.7 Portability

The website will also be portable when viewed from devices that are not computers such as mobile devices. All of the features that operate when the website opens from a computer will also operate and will be easy to use when it is opened from a device that is not a computer. This is important because as of August 2022 53.74% of all internet traffic is coming from mobile devices instead of computers [7].

1.3 Overview

Perfent will be a web-based application that recommends events to attend to groups of people.After signing in to Perfent, the users will be able to be parts of one or more groups. The users will be able to either create a group or join an existing one with invites. There can be one or more admins that

have access to special operations such as sending invites and removing members. According to the group's collective interests and their time availability along with some customized parameters that the group sets; a set of events will be presented to each group. Alternatively, the group can browse a list of all events. Other than the already existing events, the group can create custom events that they are planning to organize.

Schedules in Perfent help the group determine their common free times. Each person has a dedicated schedule for each of their groups. Using these individual schedules, Perfent creates a "group schedule." This group schedule shows the times that all the group members are available. The group schedules provide a nice way of visualizing the availability.

Perfent will gather a variety of events from popular event and ticket sites using web scraping. These events will be processed and categorized automatically. Then, based on the groups' and individuals' past event preferences and their clickstream data, new events will be recommended to the groups. These recommended events will take the group members' availability, price, distance, age, and similar preferences into account. The system will periodically recommend events to the groups. If there are more than one event that the group converges on, but they can only choose one due to time constraints or other external reasons; the members can call a vote between two or more of these events that the group wants to go to.

2 Current Software Architecture

There are other applications that are about promoting events and/or selling tickets, such as Eventbrite, Ticketmaster, Meetup, among many others. There are many similar websites, which means many different architectural styles. It is not very easy to find architectural information about applications that are not open source. However, two former Eventbrite employees (Scott Baker - former Director of Systems Engineering and Operations, Vipul Sharma - former Director of Data Engineering) describe Eventbrite's high-level software architecture in an old Quora question [8]. This discussion is not very recent, yet it provides insight to how a large event site can be constructed well. This section discusses Eventbrite's architecture and technology stack based on these two employee's answers.

According to the former Director of Systems Engineering and Operations, the site is hosted on Amazon's EC2 platform and the code is mostly written on the Django framework with Python. For load balancing and SSL encryption, they utilize haproxy/nginx. Memcached and redis are used for caching purposes. At the bottom of the architecture, there are the database systems. Hbase, MongoDB, and MySQL are used for this.

According to the former Director of Data Engineering, Eventbrite uses "Hadoop for data storage and mapreduce for processing, and hive for querying the data." Services are built mainly by using Java and Python.

A recent job listing on lever.co supports the reliability of this information [9]. Since Perfent developers are much less experienced, we decided to focus on our strengths rather than spending time learning new technologies from scratch and we came up with the architecture described in the next section. Therefore, sometimes there are similarities, sometimes there are differences.

3 Proposed Software Architecture

3.1 Overview

The proposed system architecture consists of 7 different layers, including the user interface, authentication, web server, recommender, scraper, data abstraction, and database. The web server layer manages the back-end service of the main functionalities of the system, such as event, profile, group, request, notification, and recommender management. The database abstraction layer prevents direct access to the database to prevent possible failures and uses ORM tools like Hibernate. The web scraper layer fetches and validates online events and writes them directly to the database. The database layer uses the Postgres database system and the entities. Role-based authentication is used to ensure that users can only access resources and functionalities they are authorized to access.

3.2 Subsystem Decomposition

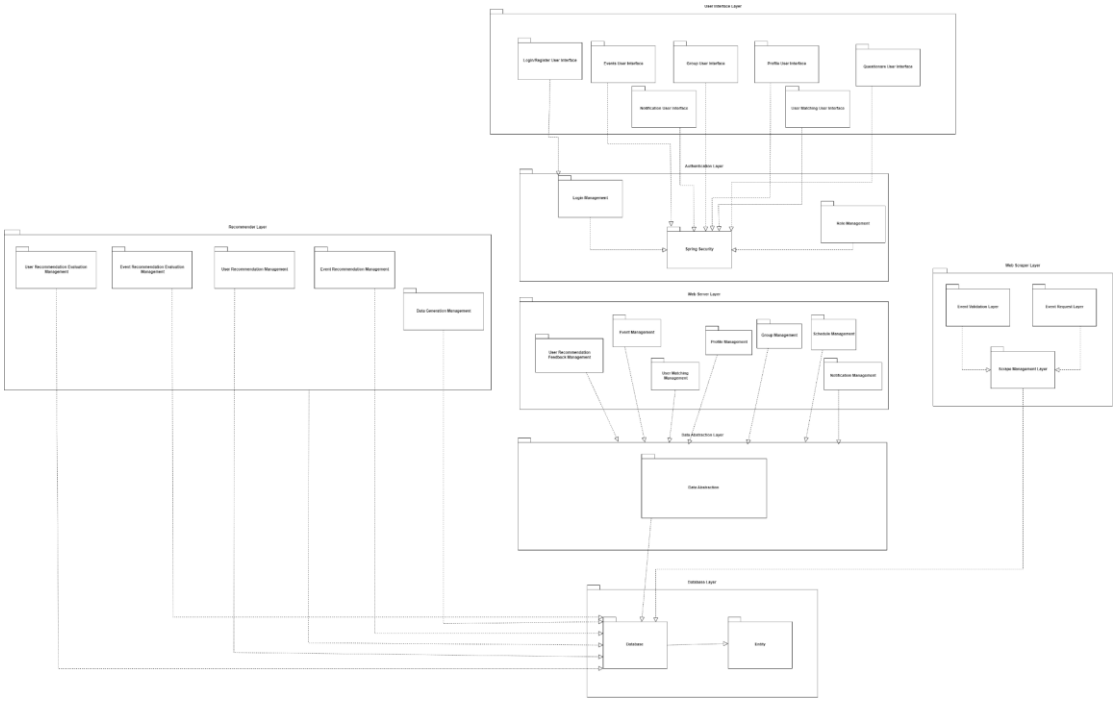

Figure 1: Subsystem Decomposition (Click the link for high resolution figure)

In this system there are 7 different layers called: user interface, authentication, web server, recommender, scraper, data abstraction and database.

In the web server layer, the back-end service of the main functionalities of the system is managed. This includes the event, profile, group, request, notification and recommender management systems. Most of these management systems communicate to the data abstraction layer to modify database entities.

In the database abstraction layer, we prevent direct access to the database to prevent possible failures. Here ORM tools such as Hibernate are used.

In the web scraper layer, the online events are fetched and validated and directly written to the database. Although direct access does not provide database protection, we realized that SQL queries perform well enough so we preferred simplicity over complexity.

Finally, in the database layer, the Postgres database system and the entities are located.

3.3 Persistent Data Management

For Perfent, data has a substantial value for accomplishing the application goal. Perfent uses the Postgres database system. First of all, the event table in this database gets updated every 15 minutes. However, when this table is updated, one of the important aspects was to avoid inserting the already existing tables or not handling invalid entries that are fetched from the web. By using the unique event id's, Perfent detects the duplicated entries and avoids reinserting them. While doing that, it also checks whether any information regarding this event has changed and if it did, performs the update operations. All of these processes are performed using SQL queries.

Secondly, when data is managed on the web server side, Hibernate is used. Hibernate is an ORM tool, which helps easy and error-proof development of database systems in Spring Boot. Because of the fact that different components of the system such as scraper and server both communicate with the same database, a possible failure there will drastically affect all of these components. From this perspective, using an ORM tool like Hibernate reduces the possible problems that can rise from raw SQL queries and makes database access easy and faster for us.

3.4 Access Control and Security

Perfent employs access control and security functionalities to ensure that the users can use the application in a safe and secure manner. First of all, to ensure that users can not reach any resources and functionalities they should not be reaching, we use role based authentication. Each role has access to a set of functionalities, and each user is given a set of roles. Users are given access to a set of functionalities according to their roles and

permissions gained through the role. Users need to use a form based login that includes email and password to use their accounts and functionalities. Users' roles can change dynamically according to operations partaken in the application, for example if a user becomes an event runner their role will be updated. Secondly, we manually check if a user is accessing a resource that they are allowed to access at each function. For example, we do not allow a user to access group information or group functionalities of a group they are not in. Finally, we use Bcrypt2 hashing to hash every password and secure their contents. In following, one can view a high level table of allowed functionalities for each role:

	Group Admin	Event Runner	Group Member	Non Group Member User	Perfent Support
Create Group	X				
View Group	X		X		
Edit Group	X				
Delete Group	X				
Create Event		X			
View Event	X	X	X	X	X
Delete Event		X			X
Edit Event		X			X
Create Group-Event		X			
View Group-Event	X		X		
Edit Group-Event	X	X	X		X
Delete Group-Event	X	X			
Create Profile	X	X	X	X	

View Profile	X	X	X	X	X
Edit Profile	X	X	X	X	
Create Notification	X				X
Edit Notification	X				X
View Notification	X	X	X	X	
Delete Notification	X				X
View Schedule	X		X	X	
Create Schedule	X			X	
Edit Schedule	X		X	X	
Delete Schedule	X			X	

Table 1: Access Control Matrix

4 Subsystem Services

Perfent's subsystems are examined in 3 parts: application, recommender, and web scraper.

4.1 Application

The application is built using the MVC (Model-View-Controller) pattern.

4.1.1 Model

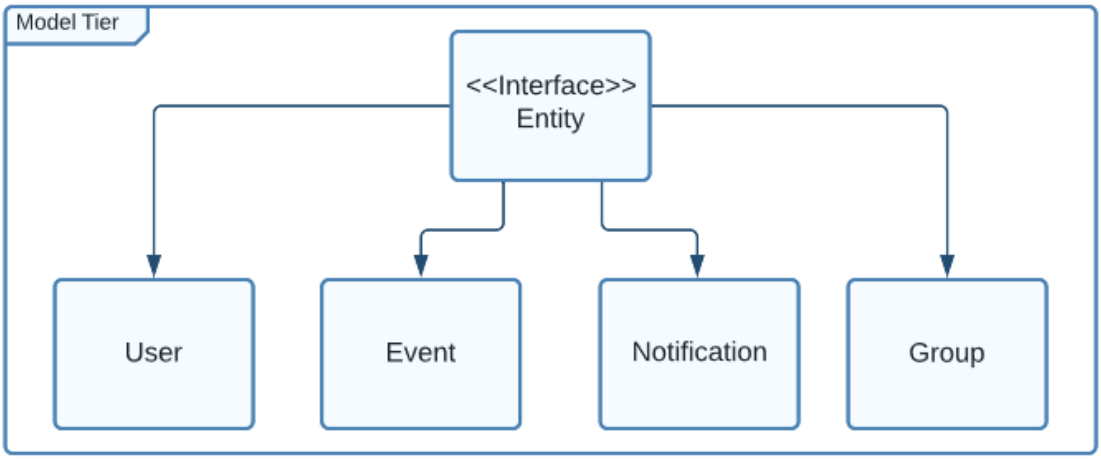

Figure 2: Model Tier

This model represents the model tier in the application layer. All entities in the application layer are implemented using Spring Boot's Entity annotation.

Class	Description
Entity <<Interface>>	The automatic entity interface of Spring Boots.
User	Entity class for application users.
Event	Entity class for events.
Notification	Entity class for notifications.
Group	Entity class for groups.

Table 2: Class Descriptions

4.1.2 View

4.1.2.1 Event Attender

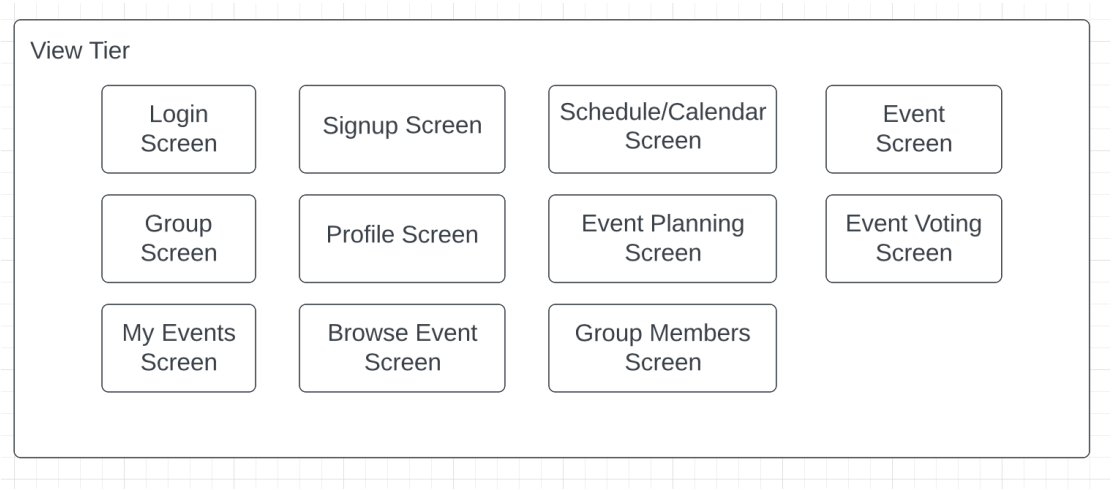

Figure 3: View Tier for Event Attender

Class	Description
Login Screen	This class provides UI components for the authentication
Signup Screen	It enables user to register to the system
Event Screen	It provides information related to the event such as date, place, etc.
Schedule/Calendar Screen	This class shows the scheduled events on the calendar of a user
Group Screen	This class provides information about the group members, group events, etc.
Profile Screen	This class shows personal information of the user
Event Planning Screen	It provides planning components for an event that group members decided to attend

Event Voting Screen	This screen enables group members to vote among events
My Events Screen	It shows the previously attended events to the user
Browse Event Screen	It provides events available on the application
Group Members Screen	This screen is for managing group members

Table 3: Class Descriptions

4.1.2.2 Event Planner

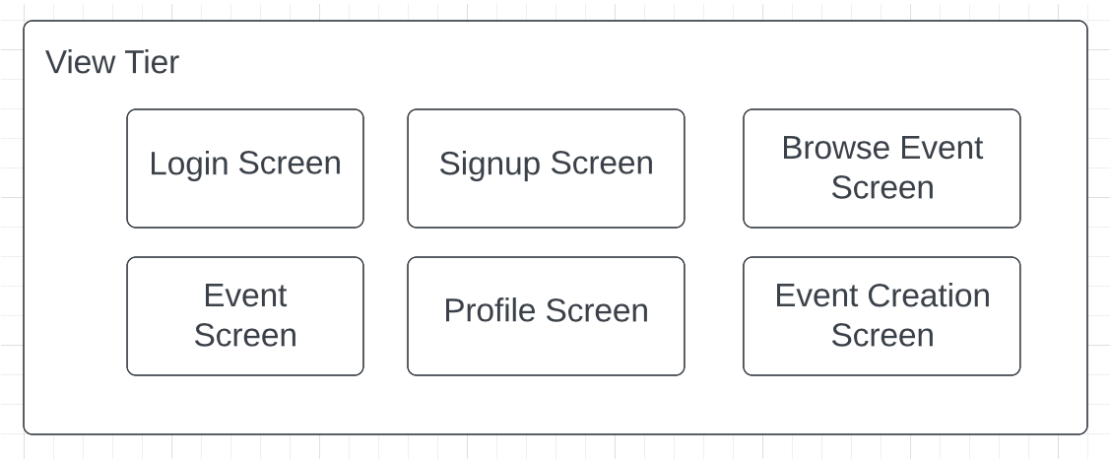

Figure 4: View Tier for Event Planner

Class	Description
Login Screen	This class provides UI components for the authentication
Signup Screen	It enables user to register to the system
Event Screen	It provides information related to the event such as date, place, etc.
Browse Event Screen	It provides events available on the application

Event Creation Screen	It enables event planner to create a new event
Profile Screen	This class shows personal information of the user

Table 4: Class Descriptions

4.1.3 Controller

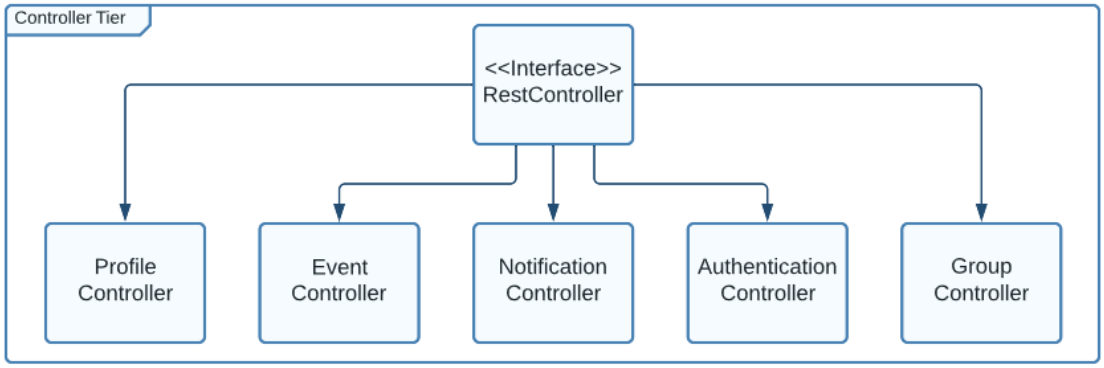

Figure 5: Controller Tier

This model represents the control tier in the application layer. All controllers in the application layer are implemented using Spring Boot's RestController annotation.

Class	Description
RestController <<Interface>>	The automatic RestController interface of Spring Boot.
Profile Controller	Controller class for application user profiles.
Event Controller	Controller class for events.
Notification Controller	Controller class for notifications.
Authentication Controller	Controller class for authentication functionalities such as login.
Group Controller	Controller class for groups.

Table 5: Class Descriptions

4.2 Recommender

The Recommender System is a crucial subsystem service within our software project that helps users discover and attend events based on their preferences. It uses various machine learning algorithms and data analytics to analyze user behavior and suggest events that match their interests. The Recommender System takes into account the user's past event attendance, click-stream data and relevant data to generate personalized recommendations.

The Recommender System subsystem service includes several components, such as a data ingestion service that collects event data, a machine learning service that trains and deploys models, and a recommendation engine that generates personalized recommendations. The system also has a feedback loop to continuously improve the accuracy of its recommendations based on user feedback.

4.3 Web Scraper

The Web Scraper is used for fetching event data from a third party web server. The data is currently obtained from “Biletix” website. The web scraping script is called from Perfent’s web server every 5 minutes and every event on the website is inserted into Perfent’s database. The events are obtained in JSON format and they are formatted before their insertion to the database.

5 Test Cases

In this section, the 50 test cases designed for Perfent are presented. Of these test cases, 36 are functional, 14 are non-functional and they are presented under their respective subsection. These test cases consist of test cases that are automated or manual; verifying/validating edge cases and/or the whole functionality. When assigning a priority/severity level, we have used the following criteria:

- Critical: The tested functionality/feature has catastrophic effects like crashing the entire site.
- Major: The tested functionality/feature prevents the users from using the application correctly.
- Minor: The tested functionality/feature has no major effects, yet it does not work as intended or is slightly annoying for the user.

5.1 Functional Test Cases

Test ID	TC#1
Test Type/Category	Functional, Usability

Title	Check if the onboarding questionnaire is shown to new users exactly once unless they open it manually from the profile
Procedure of testing steps	1. Check if the questionnaire is shown to a user who just completed the sign up process. 2. Check if the user is shown the questionnaire in the next log in after having submitted the questionnaire the first time. 3. Check if the user is shown the questionnaire in the next log in after having closed the questionnaire without submitting the first time.
Expected results	The questionnaire should only be presented immediately after signing up once or if the user wants to refill the questionnaire.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#2
Test Type/Category	Functional
Title	Verify that the user is periodically asked if they want to see more or less of the events they currently see in their event feed
Procedure of testing steps	1. Check if the “show more/less like this” question is asked at most once in every five events. 2. Check if the “show more/less like this” question is asked at least once in every twenty events.
Expected results	The question should be shown automatically once in 5-20 events in the feed.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#3
Test Type/Category	Functional, Usability
Title	Check if the available events are shown on the group schedule.
Procedure of testing steps	1. Check if at least one UI element is correctly placed for each available time range (i.e. between two busy times). 2. Check that there are no UI elements placed in unavailable times.
Expected results	If there are events that fit the group's schedule, at least one of them should be shown on the group schedule.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#4
Test Type/Category	Functional, Usability, Integration
Title	Check if the changes made on a user's Google Calendar are reflected in Perfent schedule
Procedure of testing steps	1. Check if creating events in Google Calendar is automatically replicated in Perfent schedule. 2. Check if editing events in Google Calendar is automatically replicated in Perfent schedule. 3. Check if deleting events in Google Calendar is automatically replicated in Perfent schedule.
Expected results	Every change done in Google Calendar should be reflected in Perfent individual and group schedules
Priority/Severity	Major

Date Tested and Test Result	

Test ID	TC#5
Test Type/Category	Functional, Usability, Integration
Title	Marking a time period as “busy” in Perfent should not make any changes in the user’s Google Calendar
Procedure of testing steps	1. Check if marking an available period as “busy” affects Google Calendar. 2. Check if marking an unavailable period as “busy” affects Google Calendar.
Expected results	The Google Calendar should never be affected by a change in Perfent Schedule
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#6
Test Type/Category	Functional, Usability, Safety
Title	Make sure that the names of the user’s activities in the group schedule are hidden if the user chooses to do so
Procedure of testing steps	1. Check if this user’s activities’ names are censored for everyone in the group when the user enables this option.
Expected results	When the user enables this option, their activity names must be hidden from everyone in that group.

Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#7
Test Type/Category	Functional, Safety
Title	Check if the user can see the messages of a user that they have blocked
Procedure of testing steps	1. Check if blocking a user hides the already-existing messages. 2. Check if blocking a user prevents the blocked user from messaging this user. 3. Check if blocking a user prevents this user from receiving the blocked user's messages.
Expected results	Already-existing messages should stay for both sides. The blocked user should be able to send messages, but the receiver should not see any of the new messages.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#8
Test Type/Category	Functional, Usability
Title	The user should receive a notification when they are assigned an item to bring to the event
Procedure of testing	1. Check if the user is notified in Perfent when they are assigned an item to bring to the event.

steps	
Expected results	The user should be notified when they are assigned an item to bring to the event.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#9
Test Type/Category	Functional
Title	The user should not be able to agree to an event more than once
Procedure of testing steps	1. Check if the user can agree to an event twice.
Expected results	The user should only be able to agree to an event at most once.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#10
Test Type/Category	Functional, Usability
Title	Check if recommendations are accurate to the constraints given by the group.

Procedure of testing steps	1. Determine a set of constraints to be changed. 2. In the test group, change these constraints. 3. Test if the recommendations are accurate to given constraints.
Expected results	The recommendations are accurate to given constraints.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#11
Test Type/Category	Functional, Usability
Title	Check if the recommender returns the top-matching event recommendations to the users.
Procedure of testing steps	1. In the test group, request the recommendations. 2. Compare the top recommendations from the database with responded recommendations. 3. Check if they are equal.
Expected results	Top responded recommendations should be the same as top scored recommendations in the database.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#12
Test Type/Category	Functional, Usability

Title	Check if proposed events are shown to the group at the proposed events section.
Procedure of testing steps	1. In the test group, a test group member proposes an event to the group. 2. Login to other group members' accounts in the group. 3. Check if that proposed event can be seen in the proposed events section of other members.
Expected results	Proposed events by a group member are visible by other group members.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#13
Test Type/Category	Functional, Usability
Title	Check if sorting and filtering options at event browsing are working correctly.
Procedure of testing steps	1. Determine a set of filters and sort options. 2. Choose a subset of them at each test stage. 3. Check if the chosen subset of them can accurately manipulate the shown events.
Expected results	All sort and filter options are accurately manipulating the events.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#14

Test Type/Category	Functional, Usability
Title	Check if any change on the group schedule is shown to other users in the group.
Procedure of testing steps	1. In the test group, login with a test group member and make some changes in the schedule. 2. Login with other group members. 3. Check if the first user's changes are visible for the other group members.
Expected results	Any change on the group schedule is visible to other group members.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#15
Test Type/Category	Functional
Title	Check if the user can rate an event more than once
Procedure of testing steps	1. Check if the user can rate the same event twice.
Expected results	The second rating should overwrite the first one.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#16
Test Type/Category	Functional, Usability
Title	The recommendation algorithm should not consider the user if the user marked a period of time as “not available”
Procedure of testing steps	1. Check if the recommendation algorithm recommends an event from the unavailable period when there are at least two other group members who have this period as available.
Expected results	The algorithm should disregard the unavailable user and make recommendations for other available users.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#17
Test Type/Category	Functional, Usability
Title	The different view options when browsing events should not change the order the events are presented
Procedure of testing steps	1. Check if the order of the events change when the view option is changed from “row view” to “list view” while keeping the sorting and filtering options untouched. 2. For every other view option that might be added later, check if the event order is the same as “list view” while keeping the sorting and filtering options untouched.
Expected results	The event order should not change with the same sorting and filtering options.
Priority/Severity	Minor
Date Tested and Test	

Result	

Test ID	TC#18
Test Type/Category	Functional
Title	Check if the user can be invited to a group that they are already a part of
Procedure of testing steps	1. Check if the group admin can send an invite to a group member.
Expected results	After the attempt, the group admin should receive an error message with a description.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#19
Test Type/Category	Functional
Title	Check if the user can mark an event as “attended” before the event start time
Procedure of testing steps	1. Check if the user can mark an event as “attended” before the event start time
Expected results	The user should receive an error message telling that the event has not started yet.
Priority/Severity	Minor

Date Tested and Test Result	

Test ID	TC#20
Test Type/Category	Functional
Title	The user should not be able to join to a deleted group using an old invite
Procedure of testing steps	1. Check if the user can accept an old invite from a deleted group.
Expected results	The user should receive an error message telling that the group does not exist.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#21
Test Type/Category	Functional, Usability
Title	If the only group admin leaves the group where there are more than one regular members, the oldest member should automatically become a group admin
Procedure of testing steps	1. Consider a group consisting of the members A, B, C, D where the members are sorted according to the oldest to the newest and A is the only group admin. Check if B becomes the only group admin when A leaves the group. 2. Check if the group admin is chosen randomly if there are multiple members with the exact oldest joining date by repeating the case many times.

Expected results	In such a scenario, the oldest member should become the only group admin. If there are multiple members with the same oldest joining date, the new admin should be chosen randomly.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#22
Test Type/Category	Functional
Title	Check if the user notifications are unmuted after the duration specified by the user
Procedure of testing steps	1. Check if a user who mutes their notifications for 24 hours can be notified at the 25th hour.
Expected results	The user should be notified for the new notifications after the specified unmute period ends.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#23
Test Type/Category	Functional, Usability
Title	Check if the date inputs are received by date picker UI elements

Procedure of testing steps	1. For each date input in Perfent, check if the date is submitted using a date picker.
Expected results	All dates should be submitted using a date picker unless there is an additional constraint that prevents this.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#24
Test Type/Category	Functional
Title	Check if the new users can provide invalid emails when signing up
Procedure of testing steps	1. Check if invalid emails such as "@.com" can be submitted as email addresses in the sign up screen.
Expected results	The input should only accept valid emails and it should inform the user accordingly.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#25
Test Type/Category	Functional, Safety, Usability

Title	If a user opts out of the user matching feature they are not shown to anyone and no one is recommended to them .
Procedure of testing steps	1. Unmark the option that opts in the test user to user matching feature. 2. Check if that user is recommended to any other users by checking the section that recommends users to a user. 3. Check if other test users' are recommended to the test user that has opted out of the user matching feature.
Expected results	When the user opt out of the user matching feature they are not shown to anyone and no one is shown to them.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#26
Test Type/Category	Functional, Safety
Title	Anonymously post photos or videos to the event do not contain any user information that gives the user's identity to other users.
Procedure of testing steps	1. First test user posts photo and a video to the event feed. 2. Login to the second test user and get the response that contains the data for the photo and video sent by the first user. 3. Check if that block of the response contains any user information that belongs to the first user.
Expected results	Response that contains the video and photo information from the first user, does not contain any personal user information.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#27
Test Type/Category	Functional, Usability
Title	Buttons in the application, in a short span of time can only be clicked once (no bounce effect)
Procedure of testing steps	1. For any button in the app, click the button twice in a short span of time. (Ex: 500 ms) 2. Check if the effect of the button applied twice or once.
Expected results	All of the buttons apply its effect only once when it is clicked more than one time in a short span of time.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#28
Test Type/Category	Functional, Usability
Title	Text fields in the application trims the entered texts
Procedure of testing steps	1. For any text field in the app, enter a random text in the text field with spaces present at the start and end of the text. 2. Get the value of the text field after a value is written. 3. Check if there are any empty spaces present in the fetched value.
Expected results	All text fields fetched values should be texts that are trimmed and do not contain any empty spaces at the beginning or end.
Priority/Severity	Minor

Date Tested and Test Result	

Test ID	TC#29
Test Type/Category	Functional, Usability
Title	Wishlist notifications are sent in the specified time before the event.
Procedure of testing steps	1. Add events to the wishlist of the tested user. 2. Change the events time to a date that is close to the current date. 3. Check if any notifications arrive to the tested user.
Expected results	Notification should arrive to the tested user since an event in their wishlist is close to its starting date.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#30
Test Type/Category	Functional, Usability
Title	All operations give feedback to the user whether it is an error message or success message.
Procedure of testing steps	1. For any operation in the application, do the operation in an intended and unproblematic way. 2. Check if a success message is shown. 3. Do the operation in a way that is not expected or problematic (Ex: try empty string for password) 4. Check if the error message shows up.

Expected results	For any operation an error or success message shows up.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#31
Test Type/Category	Functional, Safety
Title	Check if users can only report the users they have joined events with
Procedure of testing steps	1. Using the tested user, open the random user's section where they can be reported. 2. Do the operation that reports the randomly chosen user.
Expected results	Application should not let them report a user they have not joined events with.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#32
Test Type/Category	Functional, Usability
Title	An event or an artist should be added to the wishlist of the user when they do the operation to add them.
Procedure of testing	1. Using the test user, open the section where one can add their wanted event or artist to their wishlist.

steps	2. Click the button to add them into their wishlist. 3. Check the wishlist of the tested user to see if the chosen event or artist added to the wishlist.
Expected results	The event or the artist should appear in their wishlist.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#33
Test Type/Category	Functional, Usability
Title	Users can only use one vote in event votings.
Procedure of testing steps	1. Create a voting activity for an event for the tested group. 2. Login to a tested group member's account and try to vote more than once by clicking the vote button.
Expected results	Vote operation is not performed more than once.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#34
Test Type/Category	Functional, Usability

Title	Check if the standard expected group invite procedure is working correctly
Procedure of testing steps	1. Using the tested group admin, invite a tested non group member user to the group. 2. Login to the tested user and view the section of the application where invitations arrive. 3. Accept the invitation. 4. Login to a member of the group. 5. Check from the list of users if the invited user is now part of the group. Or 6. Reject the invitation. 7. Login to a member of the group. 8. Check from the list of users if the invited user is not added to the group.
Expected results	Invitation must be sent when the group admin sends the invitation. Invitation must be viewed in the invitation list of the invited user. If accepted, the invited user must be added to the group. If rejected, the invited user must not be added to the group.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#35
Test Type/Category	Functional, Usability
Title	Check if the group event recommendation notifications are sent periodically and not missing.
Procedure of testing steps	1. Using the tested group member, view the recommended event notification. 2. Increment the time of the system by the notification period T. 3. Check again if another event recommendation notification is sent since the system time incremented.
Expected results	Another notification is sent to the group member.

Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#36
Test Type/Category	Functional, Security, Usability
Title	Check if the standard expected login procedure is working correctly.
Procedure of testing steps	1. For the tested user, enter the correct login credentials to the text fields. 2. Check when the login button is clicked, the application lets the user login to their account. 3. Log out. 4. Enter incorrect login credentials to the text fields. 5. Check when the login button is clicked, the application does not let the user into their account.
Expected results	If correct credentials are entered, the user is taken into their account, otherwise they are not taken into their account.
Priority/Severity	Critical
Date Tested and Test Result	

5.2 Non-functional Test Cases

Test ID	TC#37
Test Type/Category	Non-functional, Accessibility

Title	Check if the web site can be accessed and used correctly from all popular browsers
Procedure of testing steps	1. Check if Perfent can be accessed and used correctly from Google Chrome. 2. Check if Perfent can be accessed and used correctly from Safari. 3. Check if Perfent can be accessed and used correctly from Edge. 4. Check if Perfent can be accessed and used correctly from Firefox. 5. Check if Perfent can be accessed and used correctly from Opera.
Expected results	Perfent should be accessed from each of these browsers and it should behave in the same way.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#38
Test Type/Category	Non-functional, Performance
Title	perfent.net should have an availability rate of at least 99% during its lifetime
Procedure of testing steps	1. Check that the (MTBF (mean time between failure)) / (total lifetime so far) is greater than or equal to 99%.
Expected results	The availability rate (the formula in step 1) should be greater than or equal to 99%.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#39
Test Type/Category	Non-functional
Title	The web scraper should be run automatically once an hour
Procedure of testing steps	1. Check if the script is run hourly on the server automatically.
Expected results	The web scraper should be run automatically once an hour
Priority/Severity	Critical
Date Tested and Test Result	

Test ID	TC#40
Test Type/Category	Non-functional
Title	The web scraper should not fetch events that already exist in the database
Procedure of testing steps	1. Check if any new event row is inserted after fetching an identical event list that has already been fetched and inserted before.
Expected results	No new rows should be inserted.
Priority/Severity	Major
Date Tested and Test	

Result	

Test ID	TC#41
Test Type/Category	Non-functional
Title	The web scraper script must complete execution in 3 minutes
Procedure of testing steps	1. Check if the execution exceeds 3 minutes.
Expected results	The execution should not exceed 3 minutes.
Priority/Severity	Minor
Date Tested and Test Result	

Test ID	TC#42
Test Type/Category	Non-functional, Usability
Title	Check if the text input fields accept Turkish characters
Procedure of testing steps	1. For each text input field in Perfent, Check if the following characters and their capital versions can be inserted and submitted: ğ, ü, ş, ı, İ, ö, ç.
Expected results	The user should be able to insert and submit these characters to text input fields unless there is a constraint that prevents this.
Priority/Severity	Minor

Date Tested and Test Result	

Test ID	TC#43
Test Type/Category	Non-functional, Usability, Performance
Title	User's clickstream data is saved to the database and not lost.
Procedure of testing steps	1. Open the events page. 2. Click and hover over events systematically according to the predefined clicking plan. 3. Check if the occurred clicking activity is written to the database.
Expected results	Occurred click and hover activity should be written to the database in some structural way.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#44
Test Type/Category	Non-functional, Security
Title	Check if session cookies are non-functional after 1 hour.
Procedure of testing steps	1. Login to the tested user's account entering the correct credentials. 2. Set the time of the system to 1 hour later. 3. Check the cookies and their active status.
Expected results	Status should be inactive and session cookie should not let the user do any more operations in their account.

Priority/Severity	Critical
Date Tested and Test Result	

Test ID	TC#45
Test Type/Category	Non-functional, Performance, Scalability
Title	Check if the server withstands the specified amount of spammed requests.
Procedure of testing steps	1. Open a load test tool. 2. Spam requests to an endpoint that does not require authorization. 3. Check if the server is still operational and answering requests.
Expected results	Server is operational unless an expected request threshold is hit.
Priority/Severity	Critical
Date Tested and Test Result	

Test ID	TC#46
Test Type/Category	Non-functional, Performance
Title	Recommendations are accurate at 70% at lowest.
Procedure of testing steps	1. Wait for the application to be used for a while or use the application with a group of people. (A human should use it) 2. Check if the result generated by the evaluation

	metrics is above 70%.
Expected results	Evaluation model gives accuracy of at least 70%.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#47
Test Type/Category	Non-functional, Security
Title	Passwords are hashed according to bcrypt2 standards
Procedure of testing steps	1. For the tested user, fetch its password in the database. 2. Compare the format of the password with bcrypt2 standards.
Expected results	Password is hashed according to bcrypt2 standards.
Priority/Severity	Critical
Date Tested and Test Result	

Test ID	TC#48
Test Type/Category	Non-functional, Performance, Scalability
Title	Any request should be responded under 1 second

Procedure of testing steps	1. Start the timer. 2. Request to the endpoint that has the highest amount of data when its request data amount and response data amount is summed. 3. Stop the timer. 4. Check if the difference between times is under 1 seconds.
Expected results	Difference is at most 1 second.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#49
Test Type/Category	Non-functional, Performance
Title	Each user's recommendations are updated after they give an explicit feedback in 1 hour.
Procedure of testing steps	1. Store the user's current recommendations. 2. The tested user rates an event or gives any other type of explicit feedback. 3. Wait for 1 hour (it may be important here to wait because of the high amounts of operations in creating recommendations) 4. Compare the current recommendations with the stored ones.
Expected results	Compared recommendations should be different.
Priority/Severity	Major
Date Tested and Test Result	

Test ID	TC#50
Test Type/Category	Non-functional, Security
Title	Check if the request sender receives an accurate error when it requests a resource their role does not have access.
Procedure of testing steps	1. Using the tested user, try to request an endpoint their role does not have access to. 2. Check if the error message saying they do not have access to that endpoint is sent as a response.
Expected results	An error message saying they do not have access to that endpoint is sent as a response.
Priority/Severity	Critical
Date Tested and Test Result	

6 Consideration of Various Factors in Engineering Design

In the development process of Perfent, our team had the mission to have an easily maintainable and valuable product in the existing market. To make our product more significant than others, we tried to consider different aspects of the engineering design.

First of all, the main viewpoint of Perfent is the contribution to the social life of humans. In the analysis stage of our application, we have considered various user profiles and how they would benefit from this application. Especially, considering the long lasting Covid-19 period, the social life of people was significantly less active than it used to be. In addition to that, scheduling events with friends has always been somewhat challenging. So to help people to have a more active social life, the requirements of Perfent are analyzed reflecting this social aspect and making social life the primary concern of the application. This was also important in terms of health because we believe social activities are significant contributors to the psychological health of humans. After the quarantine period, we believe Perfent will help people recover faster and increase their welfare.

Secondly, we tried to fetch as many events as possible with Perfent. Many events from different categories such as music, art, family, and shows are fetched from the web with regular periods. With this variety, we first tried to increase the availability for our users in terms of time and money because, in Perfent, we request our users to provide their budget. Then we recommend affordable events. Besides this economic perspective, we believe requesting

many events from different categories will also benefit event holders financially. Their events will be advertised to the correct audience due to our recommendation system and we think their customer rates will increase with this approach. In addition to these, Perfent will guide users to be updated on the events and this can potentially increase the number of cultural activities they perform as well. Eventually, we believe this will help cultural development as well.

In terms of safety, Perfent needed to consider various points. First of all, when the events are considered, we should show users events from trusted sources. Otherwise, users could have been directed to fraudulent websites. To avoid this, events are retrieved from a reliable source. Secondly, all the user information regarding the login details or preferences needs to be stored safely. For that, we utilized different security packages in the web server of the application.

Perfent is initially considered a local application specific to Turkey. Because of that, its global effects would be a consideration for now. In the future, if it fits the market in Turkey, it can be extended to other countries.

When the given details are considered, the factors and their effects can be summarized in the table below.

Factor	Effect (1-10)
Social	10
Cultural	8
Economic	7
Safety	7
Welfare	4
Public Health	3
Global	0
Environmental	0

Table 6: The factors in engineering design and their effects on Perfent

7 Teamwork Details

7.1 Contributing and Functioning Effectively on the Team

- Beste: She worked in the process of writing the reports and works actively on the web scraper and web server components.
- Çağrı: Worked in writing the reports, set up the server and applications working on the server, created the CI/CD pipeline, worked on some features of web server, and currently working on the recommendation system.
- Bora: Contributed in all reports, implementation, and brainstorming. Actively attended all group meetings and suggested ideas. Took responsibility for web scraper and web server components with Beste so far in the implementation. Reviewed code when requested.
- Faruk: Completed the required parts of the reports. Took and shared responsibility without creating any problems. Expressed his strengths and weaknesses to the team well to take the role that is the most suitable.
- Gamze Elif: Contributed all of the reports and took part in the UI design of the web-app. Currently working on the recommender system and frontend of the project.

7.2 Helping Creating a Collaborative and Inclusive Environment

- Beste: In the development process of the web scraper and web server components, she worked closely with Bora and Cagri and took their ideas. In addition to that, whenever she made a development, she used the version control system Git to create pull requests and take review them so other developers could also view the updates in the project.
- Çağrı: Suggested ideas at group meetings and showed decent contribution in the group meetings. Encouraged others to talk and voice their opinions both at live group meetings and whatsapp group chat. Talked and consulted to other group members when a problem occurred.
- Bora: Valued each member's opinions in the meetings. Always suggested ideas in a non-assertive manner in order to encourage brainstorming. Was flexible in terms of group meeting times when someone could not make it to the fixed meeting time because every member might have something valuable to add to the conversation. Encouraged the use of tools like JIRA and GitHub to make collaboration easier.

- Faruk: Shared the efforts equally with Elif while designing the front-end. Took feedback from teammates on the work that was done and changed it accordingly. Tried to make sure the work is distributed equally between the team members.

- Gamze Elif: Attended the meetings and shared ideas with team members. Most of the time shared the workload with other members and worked collaboratively on the recommender system and frontend of the project.

7.3 Taking Lead Role and Sharing Leadership on the Team

- Beste: She started the development of web scraper and web server components. After the initiation, she continued the development with other team members.

- Çağrı: Usually managed the discussions and gave direction to discussions in the live group meetings while also partaking in the discussions.

- Bora: Actively offered up ideas in subjects he is confident in. Suggested/set up meetings before regular group meetings started. Managed the use of JIRA issues. Encouraged code review tradition. Shared leadership by letting other teammates be more vocal about machine learning subjects since his machine learning knowledge is not the best.

- Faruk: Took initiative while designing and developing the front-end part of the project. Requested services from the team members that work on the back-end. Communicated with others on which parts are lagging behind and where help is needed. Participated and gave ideas in discussions during meetings.

- Gamze Elif: Give ideas about the functional requirements of the project and the implementation process. Managed the UI design of the application and also took part in implementation.

8 References

1) C. Chen, R. Alfayez, K. Srisopha, B. Boehm, and L. Shi, “Why is it important to measure maintainability and what are the best ways to do it?,” *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017.

2) I. Heitlager, T. Kuipers and J. Visser, "A Practical Model for Measuring Maintainability," 6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007), 2007, pp. 30-39, doi: 10.1109/QUATIC.2007.8.

3) B. Thorne, “Four nines and Beyond: A guide to high availability infrastructure,” *Work Life by Atlassian*, 26-Oct-2020. [Online]. Available: https://www.atlassian.com/blog/statuspage/high-availability. [Accessed: 16-Oct-2022].

4) “Where do websites store passwords?,” *The JavaScript Diaries*, 18-Nov-2019. [Online]. Available: https://www.jsdiaries.com/where-do-websites-store-passwords/. [Accessed: 17-Oct-2022].

5) C. Song, “Scalable systems 101,” *Educative*. [Online]. Available: https://www.educative.io/blog/scalable-systems-101. [Accessed: 17-Oct-2022].

6) “Does page load time really affect bounce rate? - pingdom,” *pingdom.com*. [Online]. Available: https://www.pingdom.com/blog/page-load-time-really-affect-bounce-rate/. [Accessed: 17-Oct-2022].

7) J. Gaubys, “What percentage of internet traffic is mobile? [Sep '22 UPD],” *Oberlo*. [Online]. Available: https://www.oberlo.com/statistics/mobile-internet-traffic#:~:text=As%20of%20August%202022%2C%2053.74,46.26%20percent%20coming%20from%20desktops. [Accessed: 17-Oct-2022].

8) “What is Eventbrite’s Architecture?,” *Quora*, 01-Sep-2011. [Online]. Available: https://www.quora.com/What-is-Eventbrites-architecture. [Accessed: 12-Mar-2023].

9) “Senior Software Engineer - SEO,” *Lever*. [Online]. Available: https://jobs.lever.co/eventbrite/15c7fb0c-ec37-44e8-a767-7ea818f78115. [Accessed: 12-Mar-2023].