



Bilkent University
Department of Computer Engineering

Senior Design Project
T2308
Perfent

Design Project Final 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

19.05.2023

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 CS492.

Contents

1 Introduction

2 Requirement Details

2.1 Overview

2.1.1 Groups

2.1.2 Schedules

2.1.3 Events and Recommendation

2.2 Functional Requirements

2.2.1 Event Functionalities

2.2.2 Group Functionalities

2.2.3 Schedule Functionalities

2.2.4 Profile Functionalities

2.3 Non-functional Requirements

2.3.1 Maintainability

2.3.2 Availability

2.3.3 Usability

2.3.4 Safety

2.3.5 Performance

2.3.6 Portability

3 Final Architecture and Design Details

3.1 Overview

3.2 Subsystem Decomposition

3.3 Persistent Data Management

3.4 Hardware/Software Mapping

4 Development/Implementation Details

4.1 Recommendation System

4.2 Cloud

4.3 Event Scraping

4.4 Web Server

4.4.1 Google Calendar Integration

4.5 Frontend

5 Test Cases and Results

5.1 Functional Test Cases

5.2 Non-functional Test Cases

6 Maintenance Plan and Details

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

7.2 Ethics and Professional Responsibilities

7.3 Teamwork Details

7.3.1 Contributing and functioning effectively on the team

7.3.2 Helping creating a collaborative and inclusive environment

7.3.3 Taking lead role and sharing leadership on the team

7.3.4 Meeting objectives

7.4 New Knowledge Acquired and Applied

8 Conclusion and Future Work

9 References

Design Project Final Report

Perfent

1 Introduction

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 have their individual schedules on the system where they can indicate their busy and free time slots. For groups, these schedules are combined to find a free slot for the whole group. In addition, based on the preferences of the group members, Perfent recommends 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.

2 Requirement Details

2.1 Overview

Perfent is a web-based application that recommends events to attend to groups of people. In this subsection, an overview of the features are categorized and explained.

2.1.1 Groups

After signing in to Perfent, the users are able to be parts of one or more groups. The group system is very similar to that of Whatsapp's group system [1]. The users are 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.

The users are able to join more than one group and they can "switch" between the groups so that they can see data that is related to that "current group."

The main reason a group is created is to find the best events to attend to with that group. 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, even if those events are not related to the group. If the group is interested in one of these presented or other events, they can attend that event. Detailed information about choosing an event will be explained in the following subsections.

2.1.2 Schedules

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 and they are also used to filter the recommended events, so that all the members can go to the recommended events.

In the early iterations of Perfent, the users are only able to create schedules using Google Calendar [2]. In the later stages, Perfent's own schedule interface will be implemented. Alternatively, people will be able to import calendars from different popular third-party services in the later stages.

If a time period in a user's schedule is free, but the user does not want to attend an event at that time; they can set that time period as "occupied" directly from Perfent without changing their calendar in a third-party calendar.

2.1.3 Events and Recommendation

Perfent gathers a variety of events from the popular event site Biletix using web scraping. These events are processed and categorized automatically. Then, based on the groups' and individuals' past event preferences and their clickstream data, new events are recommended to the groups. These recommended events take the group members' availability, distance, and similar preferences into account. The system periodically (once a week) recommends events to the groups.

The users do not have to wait for the system's recommendations, they can browse all the upcoming events and make suggestions to their groups as well. In Perfent's terminology, this is called "proposing an event." By using various sorting and filtering options, the users can find suitable events easily.

The events that are recommended by the system or proposed by the group members can be viewed in a list. In this list, the group members can indicate their opinions by agreeing, disagreeing, or staying neutral with each event. Perfent assumes that each group already has a platform for verbal discussion and does not complicate the implementation process by including a group chat feature. The agreeing and disagreeing provides a handy visualization for the group members which allows them to understand what the rest of the group wants. Of course, this process does not compel the

group to attend that event. This is just for understanding the group's stance on the events.

Using the feature above and understanding their stance on attending the event, the group can choose to attend an event. Then, the group admins can mark the event as "attended" indicating that they will attend the event. By doing so, the group can indicate that they are attending the event, which provides feedback to the recommendation algorithm.

In this iteration, a ticket buying or event reservation system is not implemented for events found from Biletix. However, to make things easier, Perferent will redirect the users to Biletix's event page where they can perform these actions.

The users can rate the events in order to provide additional feedback to the recommendation algorithm. Aside from the rating data, clickstream data is collected and processed to get an idea of which events draw the users attention. When recommending, the event ratings are more dominant in the recommendation choice. However, we expect a scarcity in this type of data and consequently decided to include clickstream data to help.

2.2 Functional Requirements

2.2.1 Event Functionalities

- When a new user joins the system, the system presents an optional questionnaire to gather user preferences. This questionnaire asks which categories, artists, and events the user likes from a given set.
- The users can browse all the upcoming and past events with filtering, sorting and search functionalities.
- The users can browse the events suggested to their currently chosen group.
- The users can browse the events suggested to themselves only (independent from any groups).
- The users can rate events to improve suggestion accuracy..
- The system will notify the events that are the best fit for the group periodically (once a week).

2.2.2 Group Functionalities

- The users can create groups.
- The users can join already existing groups via invitations.
- The group members can view basic information about their groups such as name, members, creation date, etc.
- The group members can propose events to their groups.
- The group admins can invite other users to their group.
- The group admin can remove members from their group.
- The group admin can make other group members group admins.

- The groups can view the events they have attended.
- The users can switch between their “group views” so that they can see what events are suggested to their currently chosen group.
- The group members can leave the groups they want.
- The group members can optionally agree or disagree to events proposed by the group members or recommended by the system.
- The group admins can mark events as attended.

2.2.3 Schedule Functionalities

- The users and groups can have a schedule view that shows events in the free periods.
- The users can mark periods in the schedule as occupied.
- The users can import an already existing schedule from Google Calendar.
- The users can synchronize their schedules if the Google Calendar schedule is updated.
- When viewing the schedules, the users can filter out other member's activities.

2.2.4 Profile Functionalities

- The users can have a profile page where other users and they can view information about the user such as name, surname, email etc.
- The users can edit some information about them such as email, password, etc.
- The users can search other user's names to find their profile pages.

2.3 Non-functional Requirements

2.3.1 Maintainability

The application's source code is mostly clean and well-organized to ease the maintenance. The application logs the possible errors to make debugging easier.

2.3.2 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. To achieve this, we use AWS's reliable hosting services and make use of their load balancers. We also installed pipelines to our GitHub repositories to automate and secure the deployment process.

2.3.3 Usability

The user interface of the application is easy to manage, simple to use, and usable. Thanks to our testers' feedback, we have ensured that all of the

pages of the user interface can be understood at a reasonable level and traversed in a maximum of 1 minute.

2.3.4 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 are hashed with BCrypt2 hashing algorithm that further protects them [3].

The data is sent over the HTTPS protocol which secures the sensitive information.

2.3.5 Performance

The application satisfies the user's waiting time expectations and prevents users from bouncing off our website. The application targets a 2-second loading time threshold with a 6-7% bounce rate for the initial (entry) loading of the website [4]. Then, for each loading of the other pages, it will target the 1-second loading time with a 6-7% bounce rate [4]. 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.

2.3.6 Portability

The website is also 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 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 [5].

3 Final Architecture and Design Details

3.1 Overview

Perfent has a compound system architecture that consists of dependent and collaborative components inside. The primary data is collected through a web scraper whereas the application resides in back-end and front-end systems. The machine learning part of the application is also a different component that regularly analyze data for better recommendation performance. In this section, the details behind this architecture and the design will be explained in further detail.

3.2 Subsystem Decomposition

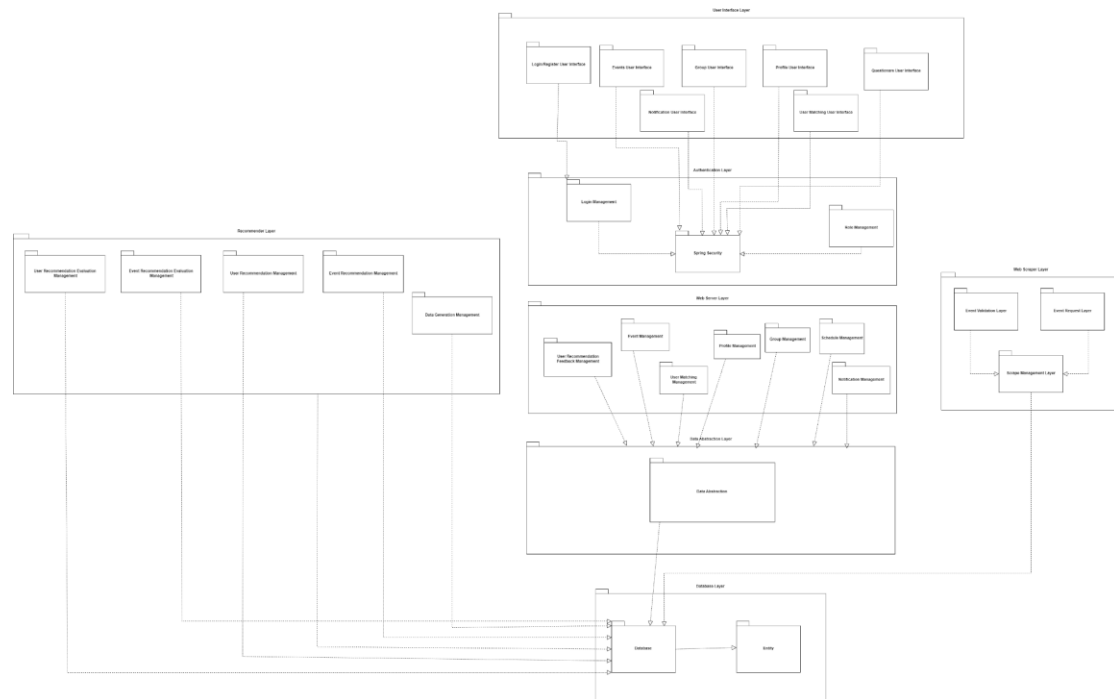


Figure 1: Subsystem Decomposition

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 Hardware/Software Mapping

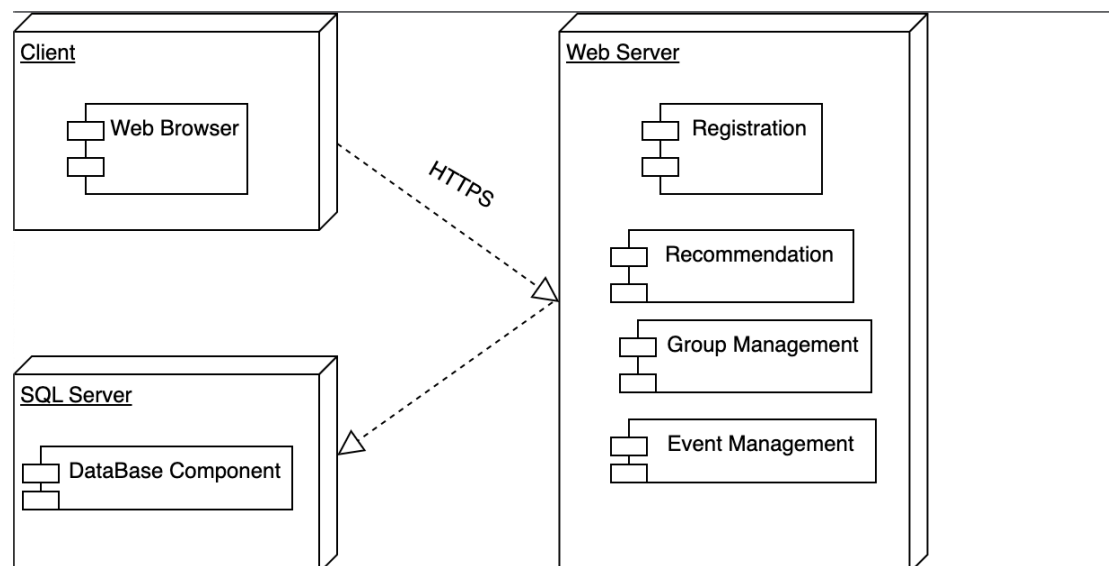


Figure 2: Hardware-Software Mapping

In the hardware-software mapping of the application, we have 3 main blocks. The client enters our system using the web browser. The web browser sends HTTPS requests to our Web server residing in AWS servers. The web server consists of different modules but it has four major modules. The first module handles the user profiles and registration process whereas the recommendation module handles the machine learning recommendation system. Group and event management modules handle the core functionalities of the system. After the HTTPS request, the corresponding module from the web server proceeds the necessary request to the SQL server. SQL server is also in AWS servers.

4 Development/Implementation Details

4.1 Recommendation System

Our recommendation system is used to recommend events for groups and individual users. The recommendation system in most general sense collects data from the user through the frontend, and processes this data to find appropriate events for the users. This recommendation system can be examined in three different subsystems: collecting user data, constructing a score system by processing this data, and recommending events. Our code operates these systems by running python scripts periodically which ensure our recommendation system adapts to the users preferences. Our code is generally working in the following principle, it fetches the data that reflects user's preferences, it matches this data with scores built in the system, and finally it finds recommended events according to scores each event received for each user.

4.1.1 Data Collection

Our recommendation system needs to collect data from the user activities to get an idea which events a user can enjoy or not enjoy. In our recommendation system we collect and use the following data:

- Our recommender collects and uses the data when a group decides to join an event.
- Our recommender collects and uses the data when a group member proposes an event to its group.
- Our recommender collects and uses the data when a group member votes in an agree/disagree voting.
- Our recommender collects and uses the data when a user clicks and browses an event.
- Our recommender collects and uses the data when a user reviews an event.
- Our recommender collects and uses the data when a user indicates their preferences on events, and categories in the onboarding form.

Through the help of frontend these data is taken from the user and sent to the backend which writes the data to the database so that our python scripts can fetch the data. These data is fetched by the recommendation system through python scripts that use normal written SQL queries with the utilization of the psycopg2 library.

4.1.2 Data Processing

Our recommendation system processes the collected data to create a score system so that our event recommendation algorithm can operate with these scores. We group the events under three groups for each user in our recommendation system.

- **Not Interacted Event:** These are the events that the user has not performed sufficient activities that the recommendation system can detect that this user has an interaction with the event. Transitioning through the **Interacted Event** group, the user is required to at least click and observe the event two times or the event should be proposed

by a group member of the user. Transitioning through the **Reviewed Event** group, users are required to indicate the system they will join the event and they review the event. Normally when the user joins the system, they have this relationship with all of the events in the system. Our prediction algorithm works on these events and predicts a score for the user.

- **Interacted Event:** These are the events that the user has shown sufficient interaction which mentioned above. We do not use the prediction algorithm on these events instead we use the data collected from the users and create a score through an evaluation framework. Transitioning through the **Reviewed Event** group, users are required to indicate the system they will join the event and they review the event.
- **Reviewed Event:** These are the events that users have joined and decided to review. We neither use the prediction algorithm or evaluation framework to calculate the scores of these types of events. Instead we directly use the user given score.

Our recommendation system requires us to use an evaluation framework and match user activities with an event interest score because we cannot use event reviews to create recommendations for the users. This is because when an event is reviewed, the event is already finished and there are no reason to recommend this event to the user.

Thus, we use the following evaluation structure to give a score for the **Interacted Events**:

- **Event Interest Score** = **Category Interest Score** (max 2 points) + **Event Proposing Score** (max 1 point) + **Event Attending Score** (max 0.75 points) + **Event Voting Score** (max 0.75 points) + **Event Browsing Score** (max 0.5 points)

Category Interest Score: Represents the interest of the user to a certain category and calculated in the following way (We might also add a time limit to the onboarding score to be active in the category score calculation. This is because user's interests might change in the future. This will be added if we notice recommendation system's performance is negatively affected by the onboarding score):

If user filled the onboarding form.

- **Category Interest Score** = Median score of all events that are not in the group of **Not Interacted Event** (max 1.4 points (normalized)) + **Onboarding Score** (max 0.6 points)

If user did not fill the onboarding form.

- **Category Interest Score** = Median score of all events that are not in the group of **Not Interacted Event** (max 2 points)

Onboarding Score: Represents the score user collected after filling the onboarding form. Onboarding score is attached to an event category.

If user's event choice is same as his/her category choice which is x:

- **Onboarding Score** of the category x = 0.6 points

If user has chosen the category x in the onboarding form:

- **Onboarding Score** of the category x = 0.48 points

If user has chosen an event with the category x in the onboarding form:

- **Onboarding Score** of the category x = 0.36 points

Event Proposing Score: Represents the score when a user proposes an event to a group.

Event Attending Score: Represents the score when a user attends to an event.

Event Voting Score: Represents the score when a user agrees or disagrees to attend an event.

If a user disagrees to attend an event.

- They get no scores from **Event Attending Score**.

If a user agrees to attend an event.

- They get full scores from **Event Proposing Score**.
- They get full scores from **Event Attending Score**.

Event Browsing Score: Represents the score of user clicking, viewing and browsing the events.

4.1.3 Predicting Event Interests

In our system, we predict the scores of events the user has not interacted with before. Thus, we predict the events grouped under the **Not Interacted Event**. To do such a prediction for a single user, we use the already calculated event interest scores of all users. To do interesting predictions we use the **Collaborative Filtering Pearson (CFP)** recommendation algorithm.

Collaborative Filtering Pearson (CFP): This algorithm is based on finding users with similar scores, and using their scores for a specific event to predict the score of another user for that specific event.

Our version of the CFP uses the pearson correlation by taking the k value 10. This means that when predicting an event's score for a user, 10 similar users are found and their scores for that specific event is used to predict the score.

4.1.4 Recommending Events

Our recommendation system recommends events in two ways: 1) recommending events to users, 2) recommending events to groups.

Recommending Events to Users: Recommending events to users is straightforward since from the other subsections above one can see that we have user event interests score for each event for each user. Thus, only thing

to do is to pick top n events when events are sorted according to their interest scores.

Recommending Events to Groups: Recommending events to groups is not straightforward as we need to recommend events in such a way that the recommended events will be liked by everyone. To achieve this we have used an aggregation method called **Average Without Misery**.

Average Without Misery: Average Without Misery is an aggregation algorithm that picks an option that has the best average score by all of the group members but on top of that it does not pick options that will be significantly disliked by some members.

Thus our version of the **AWM** algorithm picks events that have the most scores on average and on top of that picked events will not be significantly disliked by a group member.

After using **AWM** on the event interest scores of the group members, top event choices is used to get recommendations for the group.

4.2 Cloud

In cloud to host our services we use the Amazon Web Services. In order to host our backend server, frontend server, event scraper, and recommendation system we use an EC2 instance. Our EC2 instance is running on the operating system of Amazon Linux. To store and manage the data we use a PostgreSQL database which is running on RDS. To manage connections and load balance our incoming requests we use Amazon Elastic Load Balancer (ELB). Currently all these services are actively running and have no cost to us as we are running them in free tier.

4.3 Event Scraping

The implementation of event scraping was relatively easy. We sent a simple request to Biletix's site to receive a well-formatted JSON request that contained the information we needed. We run a Python script that sends this request automatically using a cron job. Once we receive the data, we convert it to our event data format and save it to our database. If a fetched event already exists, we do not save it again and if it is updated, we update it in our database.

When we started the project, we wanted to fetch data from several event sites. However, because it is a whole other problem to differentiate between the same event in different sites, we decided to move on with only Biletix events.

4.4 Web Server

We built our web server using Spring Boot with Java. We allowed the clients to send requests to our server by creating endpoints. For every functionality, we have created the necessary entity classes that represent database tables and relations. To manipulate the database contents, we created service classes that contain the application's business logic. Finally, for the clients to access these methods, we have created controller classes and methods and defined access URLs.

Throughout the implementation of the web server, we took great care of logging possible errors and handling them. If an error occurs, the client is notified with an appropriate message.

4.4.1 Google Calendar Integration

To integrate Google Calendar to Perfent, we created a project on Google Cloud Console and configured it with our domain and credential information. Then, using the Google Calendar API, we managed to get access to the users' Google Calendar activities. There were some unnecessary fields in the fetched data, so we created our own format and saved the data this way.

At the time of the implementation our site used HTTP protocol but Google Calendar API only allowed selected test users to be authorized by Google. To overcome this issue, we had to enable HTTPS for our site by getting an SSL certificate. AWS issued us a certificate and we installed the certificate to our load balancer, which enabled HTTPS. Now, anyone with a Google account can use the schedule functionalities.

4.5 Frontend

At the frontend of the application, Perfent is developed using React and MUI library. After the login process, the user credentials are used through the all front end application. This Redux pattern helped us to determine who the user is very easily.

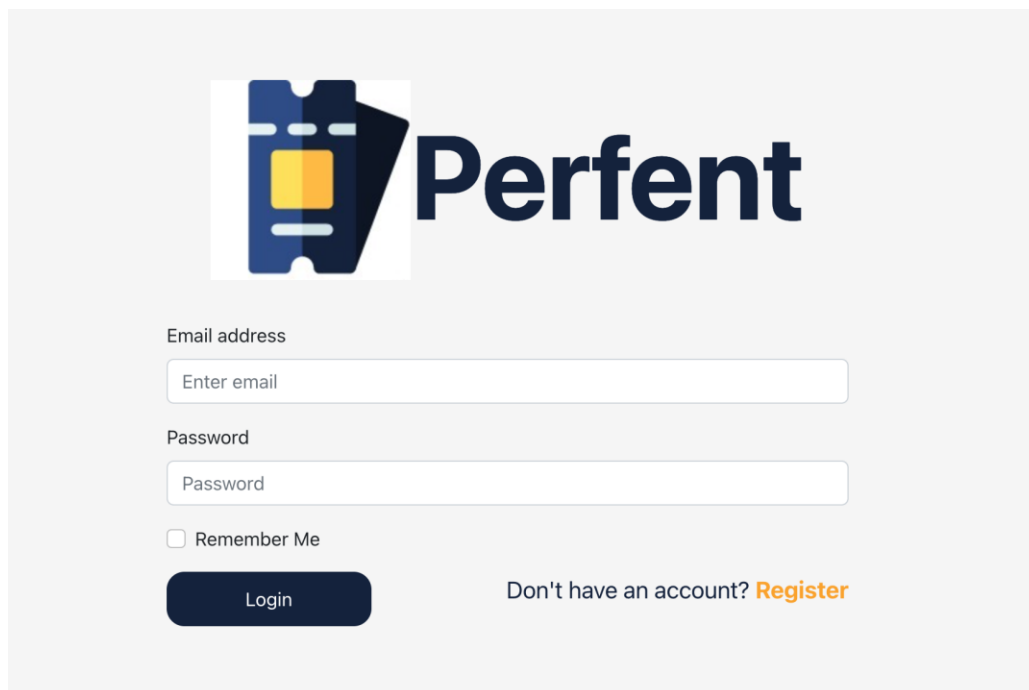

The image shows the Perfent login page. At the top, there is a logo consisting of a blue and yellow icon followed by the word "Perfent" in a bold, dark blue font. Below the logo, there are two input fields: "Email address" with a placeholder "Enter email" and "Password" with a placeholder "Password". Below the password field, there is a checkbox labeled "Remember Me". At the bottom left, there is a dark blue "Login" button. To the right of the button, there is a link that says "Don't have an account? Register" where "Register" is in orange.

Figure 3: Perfent Login Page



Perfent

Name

Email address

Password

Confirm Password

[Register](#) [Already have an account? Login](#)

Figure 4: Perfent Signup Page


Welcome To Perfent

Perfent aims to find the perfect events for you. To be able to do that, we would like to ask a few questions about your event preferences.

[MOVE TO QUESTIONNAIRE](#) [MOVE TO HOMEPAGE](#)

Figure 5: Perfent Questionnaire Page

At Perfent, to understand the user preferences better, we ask for an optional questionnaire upon registration. The choices in this questionnaire, provides better suggestions for users.



Question 1/4

So, let's start with people. Choose your favorite artists below!

☐ Tarkan
 ☐ Beren Saat
 ☐ Hande Yener

☐ Sıla
 ☐ Cem Yılmaz
 ☐ Kenan İmirzalıoğlu

☐ Kıvanç Tatlıtuğ
 ☐ Hazal Kaya

☐ Bergüzar Korel
 ☐ İbrahim Tatlıses

☐ Gülben Ergen
 ☐ Sertab Erener

☐ Sezen Aksu
 ☐ Barış Arduç



Question 2/4

What about choosing your favourite categories?

☐ MUSIC
 ☐ SPORT
 ☐ ART
 ☐ FAMILY



Question 3/4

Let's dive further in categories

☐ POP
 ☐ ROCK
 ☐ JAZZ
 ☐ ALTERNATIF
 ☐ KLASİK

☐ BALE_DANS
 ☐ TIYATRO
 ☐ GOSTERI

☐ STAND_UP
 ☐ BASKETBOL

☐ VOLEYBOL
 ☐ FUTBOL
 ☐ MOTORSPORT

☐ SİRK
 ☐ MUSICAL
 ☐ THEMEPARK
 ☐ ZOO

☐ ATTRACTIONCENTER



Question 4/4

Last question, what is the best day to party?

☐ Monday
 ☐ Wednesday
 ☐ Tuesday
 ☐ Thursday

☐ Friday
 ☐ Sunday
 ☐ Saturday

←

SUBMIT ANSWERS →

Figure 6: Perferent Questionnaire Pages

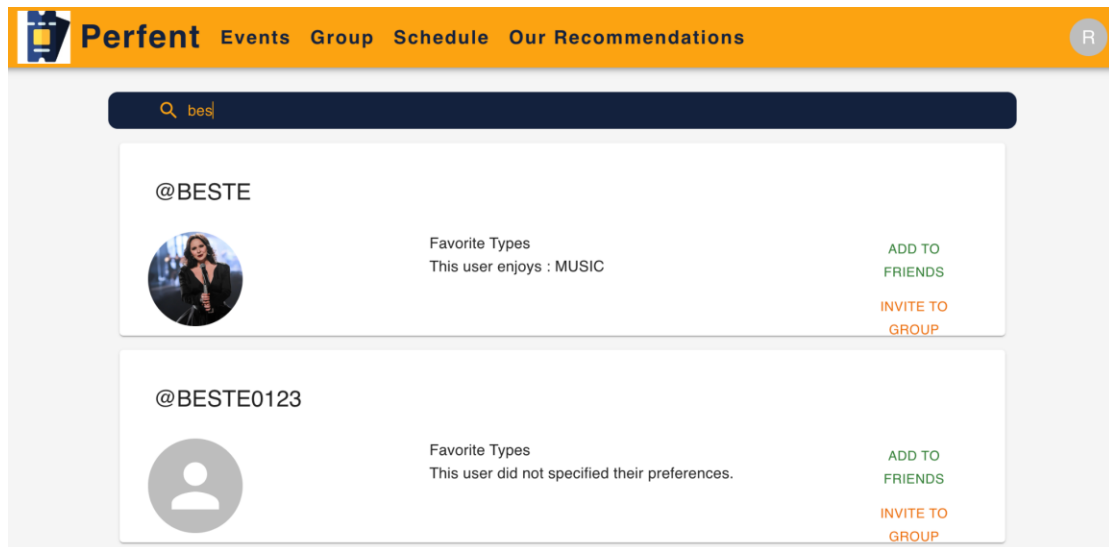


Figure 7: Perfent User Search Page

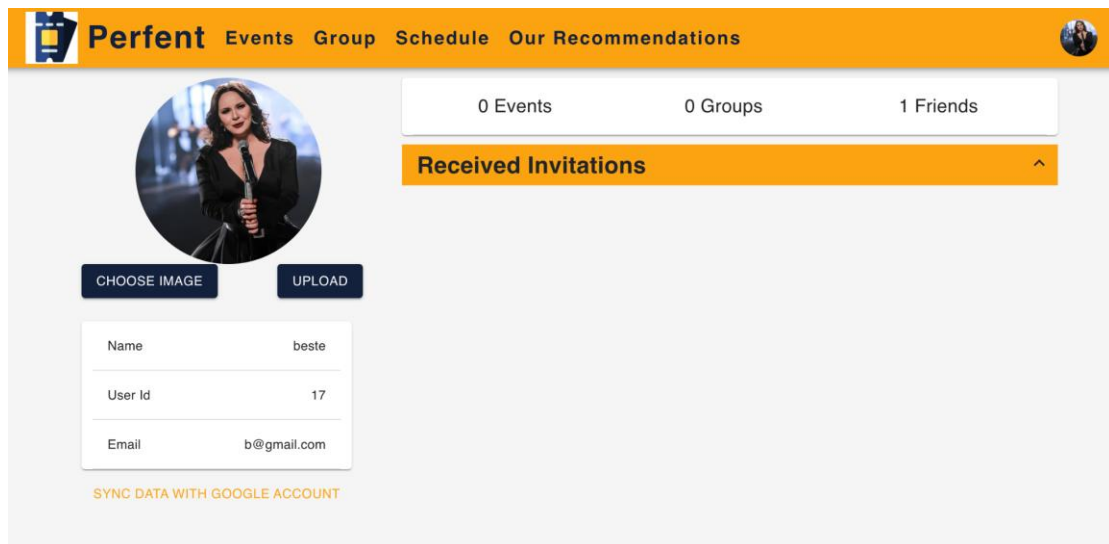


Figure 8: Perfent User Profile Page

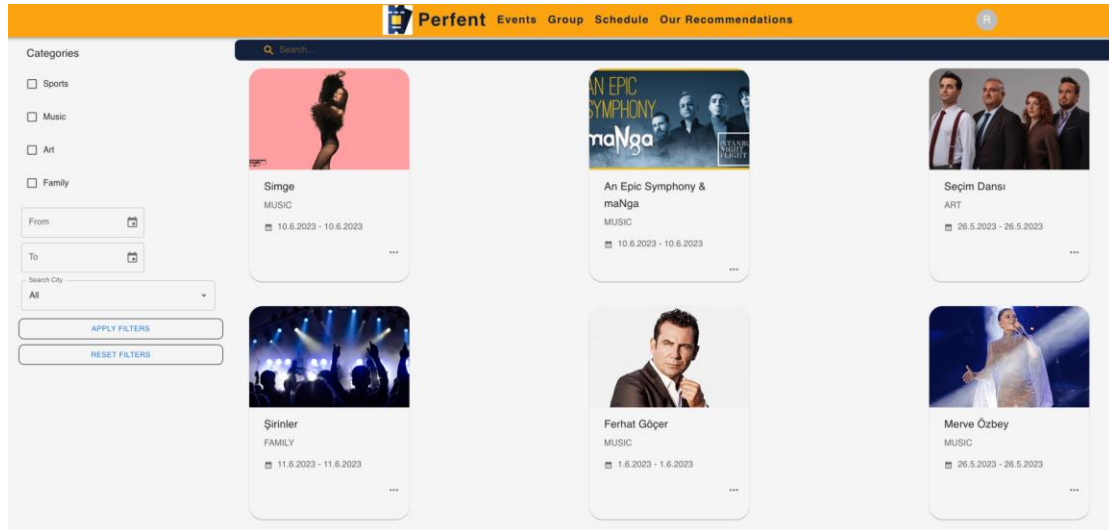


Figure 9: Perfent Events Page

5 Test Cases and Results

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	N/A

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	17.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	N/A

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	<ol style="list-style-type: none"> 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	N/A

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 steps	<ol style="list-style-type: none"> 1. Check if the user is notified in Perfent when they are assigned an item to bring to the event.
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	N/A
------------------------------------	-----

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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Partially Passed

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	<ol style="list-style-type: none"> 1. In the test group, login with a test group member and make some changes in the schedule.

steps	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	19.05.2023 / Passed

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	18.05.2023 / Passed

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	N/A

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

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	18.05.2023 / Passed

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	18.05.2023 / Passed

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	18.05.2023 / Failed

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

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	N/A

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	18.05.2023 / Passed

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	19.05.2023 / Passed

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	1. Unmark the option that opts in the test user to user matching feature.

steps	<ol style="list-style-type: none"> 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	N/A

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	<ol style="list-style-type: none"> 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	N/A

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	<ol style="list-style-type: none"> 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	19.05.2023 / Passed

Test ID	TC#28
Test Type/Category	Functional, Usability
Title	Text fields in the application trims the entered texts
Procedure of testing steps	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	N/A

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	<ol style="list-style-type: none"> 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	18.05.2023 / Partially Passed

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	<ol style="list-style-type: none"> 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	N/A

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 steps	<ol style="list-style-type: none"> 1. Using the test user, open the section where one can add their wanted event or artist to their wishlist. 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	17.05.2023 / Passed

Test ID	TC#33
Test Type/Category	Functional, Usability
Title	Users can only use one vote in event votings.
Procedure of testing steps	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

Test ID	TC#34
Test Type/Category	Functional, Usability
Title	Check if the standard expected group invite procedure is working correctly
Procedure of testing	<ol style="list-style-type: none"> 1. Using the tested group admin, invite a tested non group member user to the group.

steps	<ol style="list-style-type: none"> 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. <p>Or</p> <ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	17.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	15.05.2023 / Passed

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	<ol style="list-style-type: none"> 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

	<p>from Edge.</p> <ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	18.05.2023 / Passed

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	18.05.2023 / Passed

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	18.05.2023 / Failed (takes ~7 minutes)

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	17.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	17.05.2023 / Passed

Test ID	TC#46
Test Type/Category	Non-functional, Performance
Title	Recommendations are accurate at 70% at lowest.
Procedure of testing steps	<ol style="list-style-type: none"> 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	N/A
------------------------------------	-----

Test ID	TC#47
Test Type/Category	Non-functional, Security
Title	Passwords are hashed according to bcrypt2 standards
Procedure of testing steps	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

Test ID	TC#48
Test Type/Category	Non-functional, Performance, Scalability
Title	Any request should be responded under 1 second
Procedure of testing steps	<ol style="list-style-type: none"> 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	18.05.2023 / Failed (1% failure rate - doesn't exceed 3s)

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

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	<ol style="list-style-type: none"> 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	18.05.2023 / Passed

6 Maintenance Plan and Details

All of our systems are currently working on the Amazon Web Services. AWS provides powerful tools for us to monitor and get a good idea about the status of the services they provide. For example we can observe the EC2 instance's (where we run our virtual machine) CPU utilization (Figure 10), and incoming traffic (Figure 11). We plan to use these types of tools which are present in all AWS services to perform a good maintenance of the project.

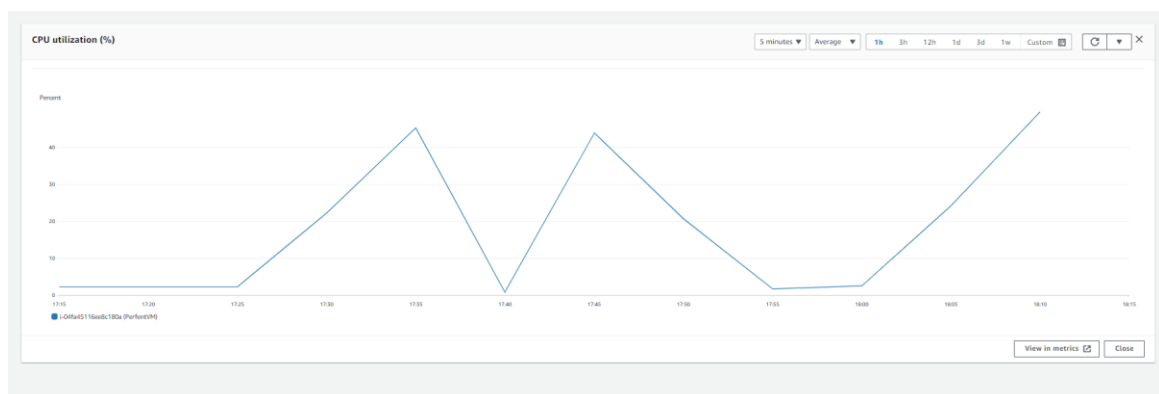


Figure 10: CPU Utilization Graph

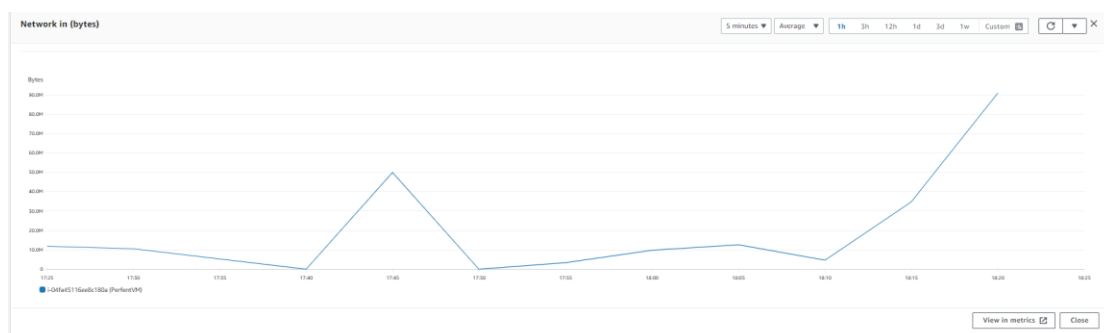


Figure 11: Network Traffic Graph

In places where we find our services lacking and insufficient for our goals we can easily upgrade our services since AWS offers the capability to do so. For example, through using the following user interface available in RDS (Figure 12) we can add additional space to our database and improve the performance of our services.

Figure 12: Storage Allocation

7 Other Project Elements

7.1 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 1: The factors in engineering design and their effects on Perfernt

7.2 Ethics and Professional Responsibilities

The schedule information provided by the users, or any other confidential information is not shared with any third party company unless the user agrees to share. Sensitive user information such as user passwords and locations are stored securely. To achieve this, the data is encrypted before they are saved to the database.

A positive working environment was needed for the team to work in harmony. To achieve this, the developers respected each other, the work was shared as evenly as possible, and the developers were transparent with their progress during the weekly progress meetings.

7.3 Teamwork Details

7.3.1 Contributing and functioning effectively on the team

- 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 in the implementation. Reviewed code when requested. Enabled HTTPS. Helped generate synthetic data for testing purposes.
- Beste: She worked in the process of writing the reports and works actively on the web scraper and web server components. In addition to that, she developed certain pages at front end and helped in machine learning parts as well.
- 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. Reviewed other's pull requests on the front-end.
- Elif: Collaborated with the team members for the reports. Worked on several components for the frontend and contributed mostly on fetching data from backend and displaying the data when needed. Also added ui analytics integration for gathering user data for recommender system.

- Çağrı: Worked in all reports, and implementation. Actively attended all group meetings and suggested ideas. Worked on the recommendation system, authentication and on cloud services.

7.3.2 Helping creating a collaborative and inclusive environment

- 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.
- 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.
- 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.
- 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. Take the opinions of the other team members' into consideration and meet their requests throughout the implementation process.
- 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.

7.3.3 Taking lead role and sharing leadership on the team

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

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

7.3.4 Meeting objectives

- Bora: Met most of the objectives of the project. All of the reports were completed successfully and on time. These were approved by the course instructor and our supervisor. For the implementation objectives, he was responsible for the web-server, web scraper, some integration parts and the testing of the corresponding parts. The scraper was handled well as a team. The web server could not be completed 100%, yet it was mostly completed. The corresponding integration objectives were handled well as well. Unfortunately, he could not complete the verification & validation objectives well enough. Only the amount necessary to deliver other parts were completed due to the time constraints.
- Beste: She met most of the objectives in the project. She learned API development with Spring and database design and implementation with PostgreSQL and Hibernate. In addition to that, for web scraping and recommendation system she developed new skills in Python. She completed all her main goals in project plan which are implementation of server and scraper. She took part in the integration of web server and scraper and the integration of front end and web service. She additionally helped other team members actively.
- Elif: Learned how to integrate external ui components and libraries to the frontend design using React.js. Improved existing skills regarding frontend and backend integration. Contributed recommendation system for obtaining data.
- Faruk: During implementation used React-MUI library for the first time to create front-end components. Implemented the connection between front-end and back-end for some parts. Worked on reports to help the team achieve success during both semesters.
- Çağrı: Completed the main goal of building a recommendation system. Although, if there were more time there were plans to combine the recommendation algorithms in a way that can be more effective. Also, completed the main goal of building a cloud environment where Perfent

services can be hosted. Other than these two, there were other minor objectives I have completed.

7.4 New Knowledge Acquired and Applied

First of all, team members learned how to use cloud systems. By using AWS, we deployed our application. This helped us to learn how to maintain and improve a real time application. For that, CI/CD pipeline has been effectively used by our team.

Our team members developed themselves in different areas. The members of the backend team learned the efficient design of database and API whereas our frontend team improved themselves in user friendly UI designs and implementing fast web applications. All of the group members improved themselves on reading documentations and adapting to new technologies. As a result, web technologies such as Spring and React have been learned by the team. Lastly, some of the team members developed their skills on data processing and machine learning through the development of recommendation system.

8 Conclusion and Future Work

As of now Perfent is a web application that allows people to browse events with their group using basic functionalities. To improve the application the group's aim is to perfect these functionalities before adding new ones and explore new options. The possible improvements are as follows:

Optimizing the recommender: In order to give better recommendations the recommender needs more data from the users. This will only happen when the application is more widely used. Therefore, we think that with more users the recommender will get better and attract more users.

Launching a mobile application: For modern web applications such as Perfent, a mobile counterpart is necessary for achieving wide usage.

Scraping events from more sites: Right now Perfent only uses events that are posted on Biletix website. However, there are more sources that can be used especially if events from other countries will be used.

Expanding to different countries: If success in the Turkish market is achieved an expansion to other countries is inevitable.

9 References

- [1] "Features," Whatsapp. [Online]. Available: <https://www.whatsapp.com/features>. [Accessed: 13-Nov-2022].
- [2] "Calendar," Google Workspace. [Online]. Available: <https://workspace.google.com/products/calendar/>. [Accessed: 13-Nov-2022].

- [3] "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].
- [4] "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].
- [5] 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].