Assignment - Web Programming

This project was done with fellow uni student

Section 1: Application requirements

The objectives to be achieved through the implementation of the application are:

- Informing users about upcoming and past events
- Ability to browse upcoming events and view their details
- Ability to purchase tickets for event screenings
- Seat selection capability
- Event search functionality
- Creating a user-friendly website
- Creating an administrator page for easy and efficient application management

Work Implementation Phases:

Phase 1: General idea capturing of the assignment (Late May)

Phase 2: Connecting the Application to the MongoDB database (June 1st - June 5th)

Phase 3: Adding initial pages such as homepage, login, etc. (June 5th - June 12th)

Phase 4: Adding specific pages and functionalities (June 12th - June 20th)

Phase 5: Refining functionalities, bug fixing, and creating a report (June 20th - June 25th)

Questionnaire: https://forms.gle/bwwiQVo8g1m3ehcr6

Answer Sheet:

https://docs.google.com/spreadsheets/d/1WcYVFnGXgaxxgbCNbbCDSX7VoRqn41gxJUb-GnMkes8/edit?resourcekey#gid=1563126204

Google Forms and Google Docs were used to create the questionnaire and document the results.

User Personas

1. Name: Haris

Age : 32

Background: Haris enjoys gaming and is interested in gaming events during his free time. As an event attendee, he appreciates the option to choose his seat during screenings. He also has a favorite event that he wants to attend regularly and plans for it.

Goals: Haris's main goal is to easily and quickly find his favorite event and secure his preferred seat. He desires a user-friendly website that enables him to accomplish these tasks.

Behavior: Haris regularly checks the website for updates on his favorite event and others. He also likes to browse event details to see if there are any similar events he might be interested in purchasing tickets for.

2. Name: Sofia

Age : 25

Background: Sofia is a popular influencer in the gaming community and creates content related to gaming events. She has a large following in various gaming communities. Sofia shares her experiences and live streams from the events she attends. She is always eager to explore new events and share them with her audience.

Goals: Sofia wants to continuously discover new events and have easy guidance to attend them. She enjoys going to events with her friends and collaborators if necessary. Additionally, she wants to collaborate with event organizers and promote the events to her followers.

Behavior: Sofia spends a significant amount of time on social media platforms, seeking information and interacting with the gaming community. She prefers a visually appealing website where she can track all upcoming events, view event details, and select an extra type of ticket for herself and her friends.

3. Name: Alex

Age: 19

Background: Alex is a passionate gamer who enjoys attending gaming events and staying informed about the latest trends in the video game industry. He has attended several events in the past and is always looking for new ones to attend.

Goals: Alex wants to discover upcoming gaming events, be able to view information about these events, and purchase tickets for the events that interest him.

Behavior: Alex dedicates significant time to online gaming research. He appreciates user-friendly websites that are easy to navigate.

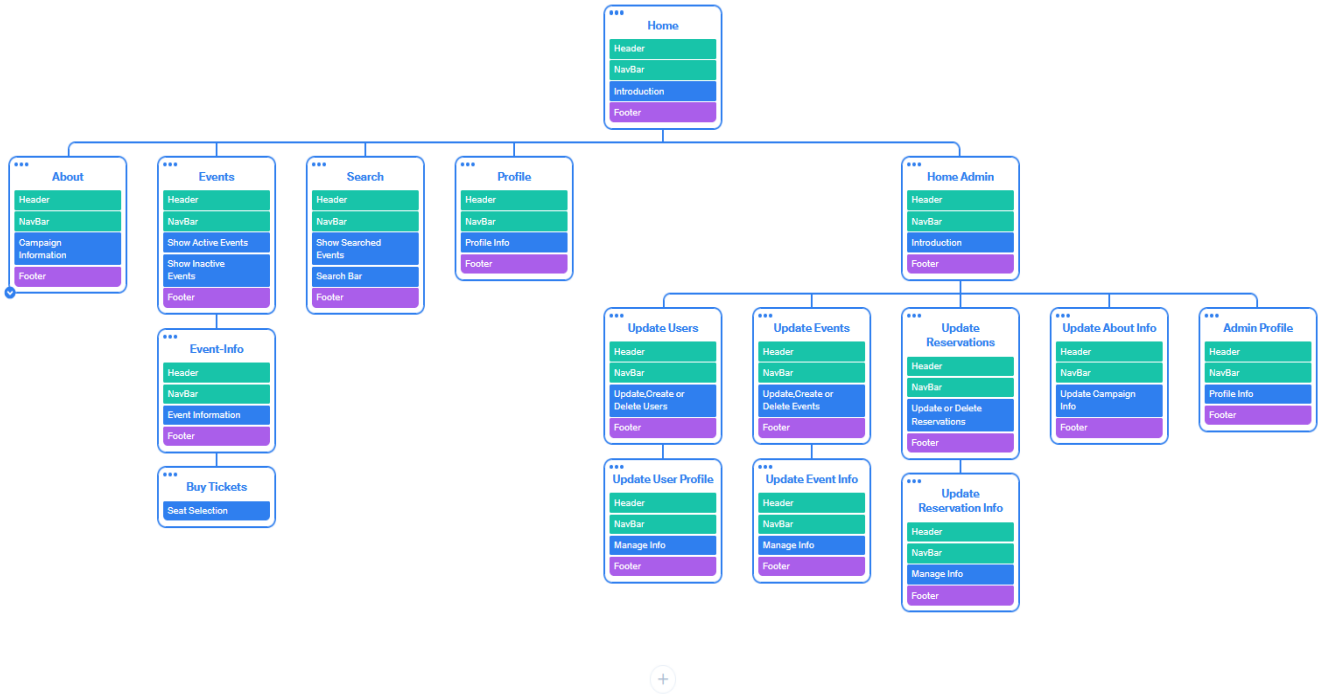
Frustrations: Missing out on events due to lack of information.

User Stories

1. As a user, I want to be able to sign up/log in to the application.
2. As a user, I want to navigate the pages easily, both on the computer and on mobile/tablet devices.
3. As a user, I want to have access to my profile and be able to edit it.
4. As a user, I want to see general information about the campaign.
5. As a user, I want to view scheduled events of the campaign, including completed and upcoming events.
6. As a user, I want to be able to select an event and view more information about it.
7. As a user, I want to see the location of the event on a map.
8. As a user, I want to make a reservation for a presentation at the events.
9. As a user, I want to be able to choose the type of ticket and the number of tickets.
10. As a user, I want to search for an event based on its name.
11. As a user, I want to easily access the campaign's social media platforms.
12. As a user, I want the option to switch the page's appearance to dark mode.
13. As an administrator, I want to have access to the admin dashboard and user profiles.
14. As an administrator, I want to be able to log in and log out from any page.
15. As an administrator, I want to have access to a page that displays all users, where I can modify their details, create new users, and delete users.
16. As an administrator, I want to have access to a page that displays basic campaign information, where I can modify and delete information.
17. As an administrator, I want to view all events, view their details, and have the ability to add, modify, or delete information.
18. As an administrator, I want to view all reservations, view reservation details, and have the ability to modify or delete reservations.

Visual Sitemap, Wireframe, mockup:

Visual sitemap:

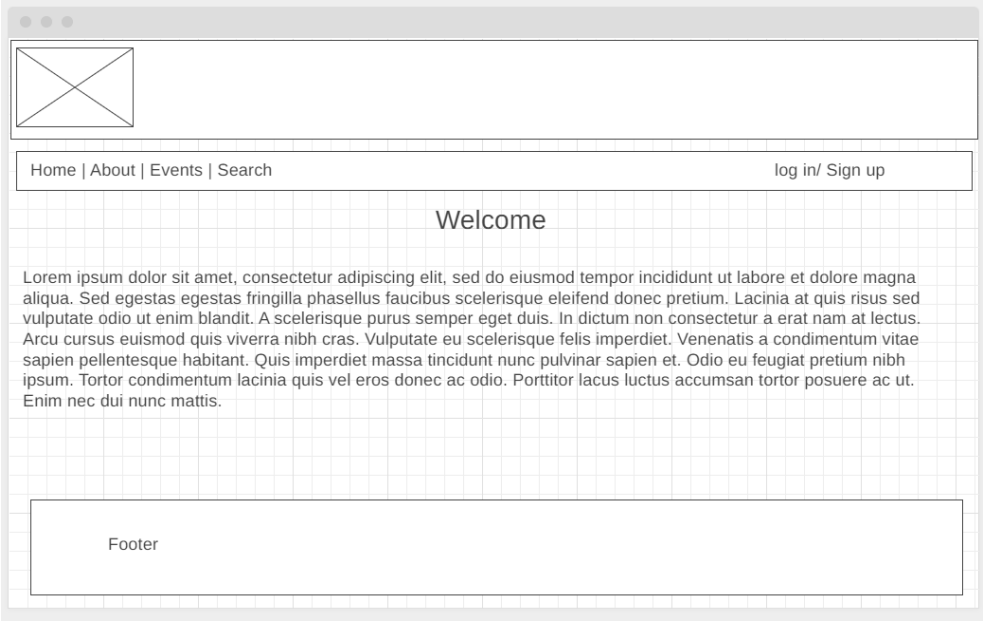


To create the sitemap, the following website was used:

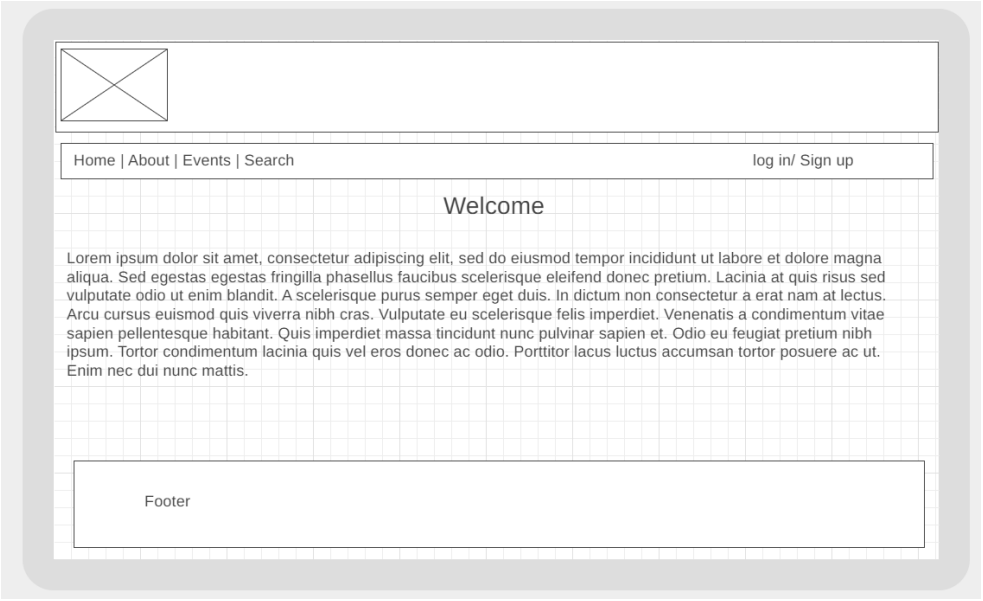
https://octopus.do/l4mzl3wg4lo

Wireframe:

Desktop:

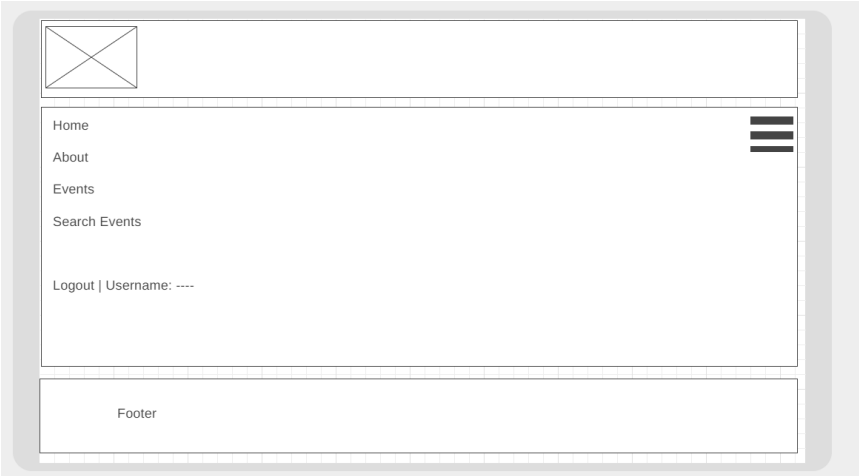


Tablet:



Smartphone:

With the selection that will open the dropdown menu in the top navigation bar:

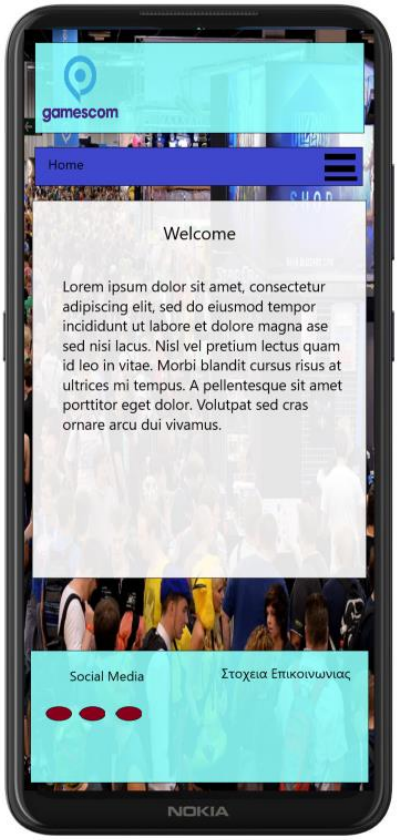


Mockup:

Desktop

Tablet

Smartphone:

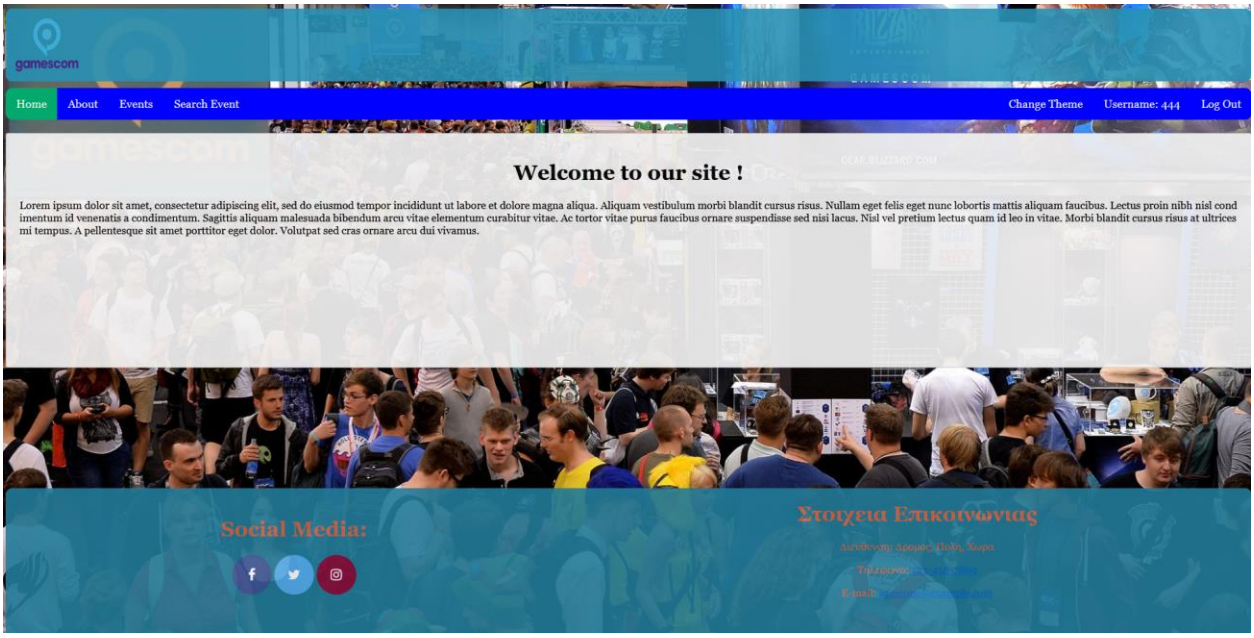


Section 2: Modern Trends in Web Applications

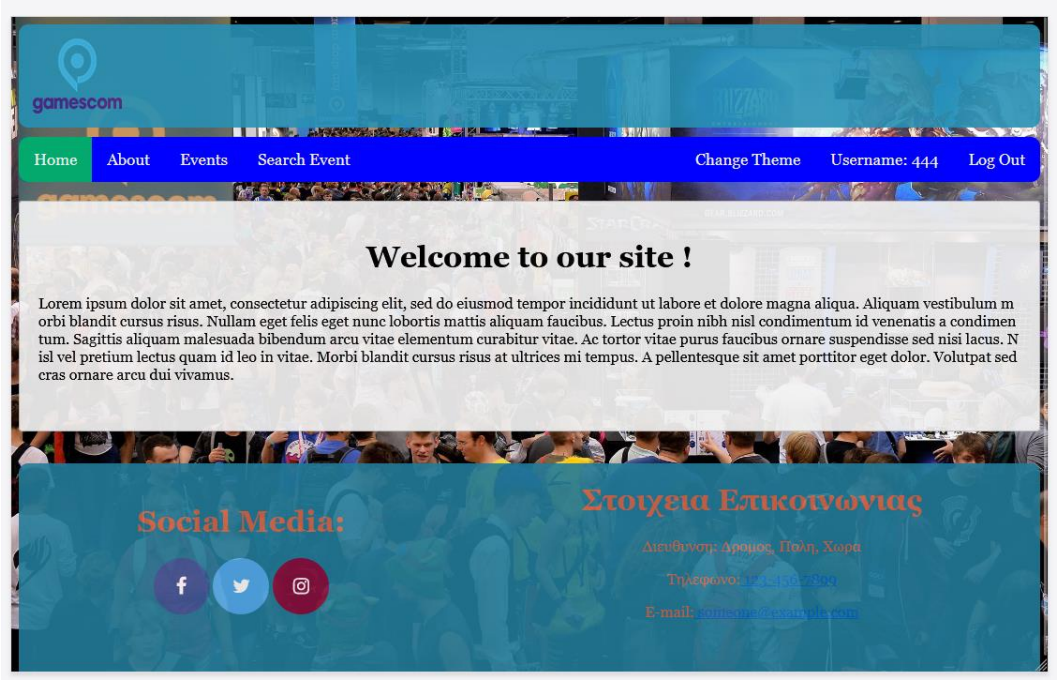
In our project, we have utilized three modern trends in web applications for 2023: Dark Mode, JavaScript Framework, and Responsive Design.

Responsive Design: On the homepage, when the application is opened on different devices or the page layout is rearranged from desktop, the elements on the page will adjust their size accordingly.

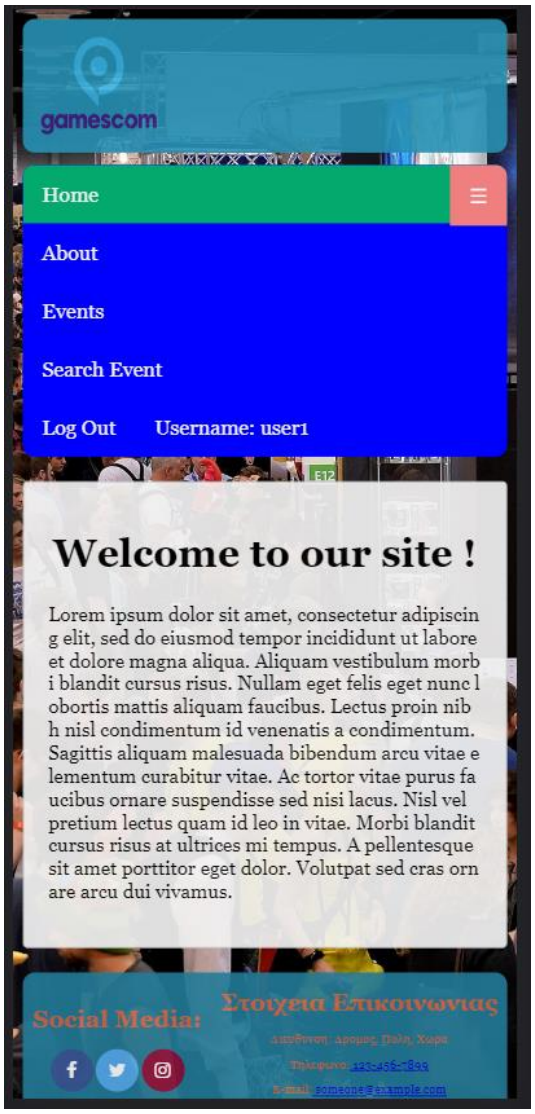
Desktop:



Tablet:

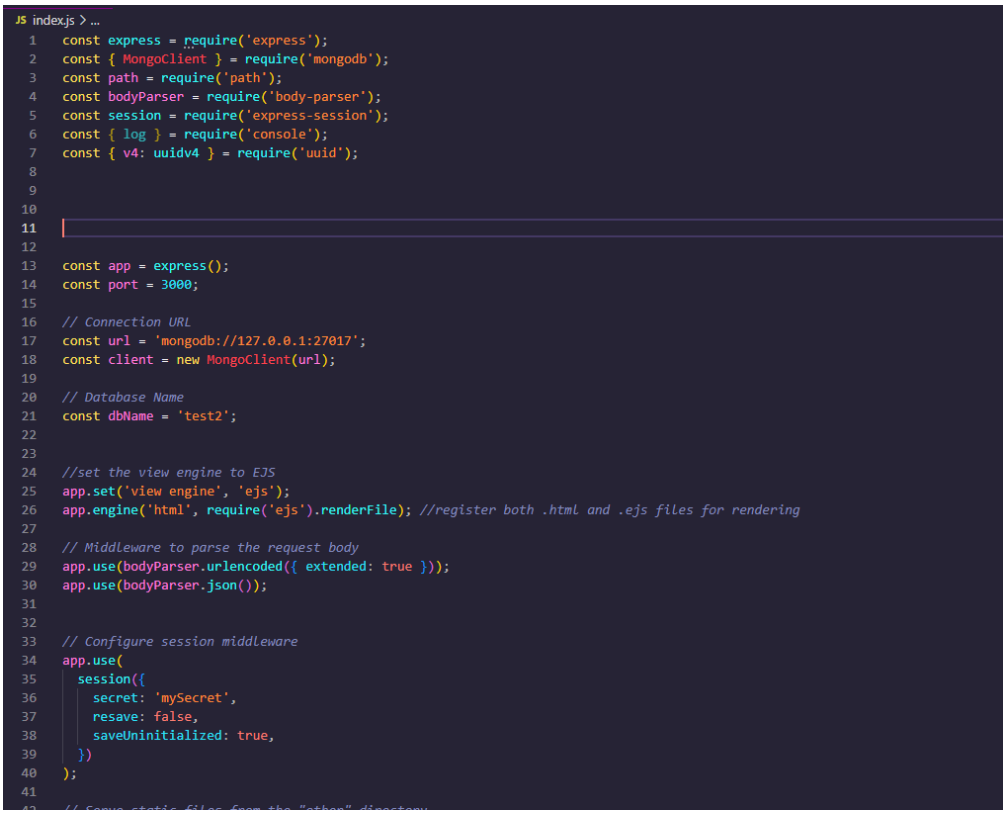


Smartphone:


gamescom
Home
About
Events
Search Event
Log Out
Username: user1
Welcome to our site !
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Aliquam vestibulum morbi blandit cursus risus. Nullam eget felis eget nunc lobortis mattis aliquam faucibus. Lectus proin nibh nisl condimentum id venenatis a condimentum. Sagittis aliquam malesuada bibendum arcu vitae elementum curabitur vitae. Ac tortor vitae purus faucibus ornare suspendisse sed nisi lacus. Nisl vel pretium lectus quam id leo in vitae. Morbi blandit cursus risus at ultrices mi tempus. A pellentesque sit amet porttitor eget dolor. Volutpat sed cras ornare arcu dui vivamus.
Social Media:
Στοιχεια Επικοινωνιας

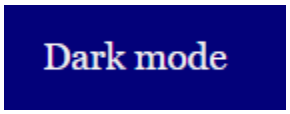

gamescom
Home
Welcome to our site !
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Aliquam vestibulum morbi blandit cursus risus. Nullam eget felis eget nunc lobortis mattis aliquam faucibus. Lectus proin nibh nisl condimentum id venenatis a condimentum. Sagittis aliquam malesuada bibendum arcu vitae elementum curabitur vitae. Ac tortor vitae purus faucibus ornare suspendisse sed nisi lacus. Nisl vel pretium lectus quam id leo in vitae. Morbi blandit cursus risus at ultrices mi tempus. A pellentesque sit amet porttitor eget dolor. Volutpat sed cras ornare arcu dui vivamus.
Social Media:
Στοιχεια Επικοινωνιας

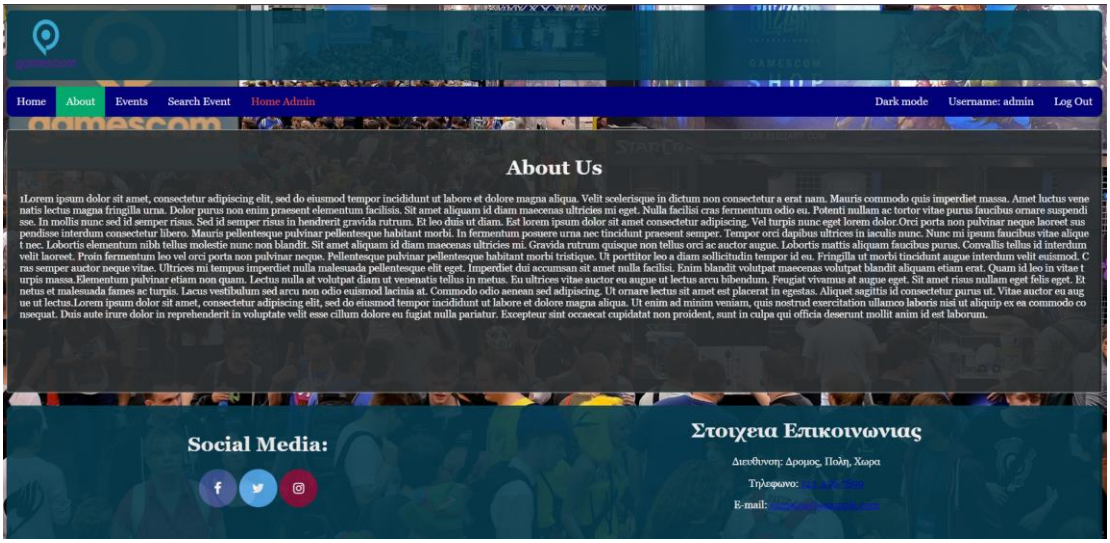
JavaScript Framework: We used Node.js and Express.js to implement the server for our application.



```
JS index.js > ...
1   const express = require('express');
2   const { MongoClient } = require('mongodb');
3   const path = require('path');
4   const bodyParser = require('body-parser');
5   const session = require('express-session');
6   const { log } = require('console');
7   const { v4: uuidv4 } = require('uuid');
8
9
10
11
12
13  const app = express();
14  const port = 3000;
15
16  // Connection URL
17  const url = 'mongodb://127.0.0.1:27017';
18  const client = new MongoClient(url);
19
20  // Database Name
21  const dbName = 'test2';
22
23
24  //set the view engine to EJS
25  app.set('view engine', 'ejs');
26  app.engine('html', require('ejs').renderFile); //register both .html and .ejs files for rendering
27
28  // Middleware to parse the request body
29  app.use(bodyParser.urlencoded({ extended: true }));
30  app.use(bodyParser.json());
31
32
33  // Configure session middleware
34  app.use(
35    session({
36      secret: 'mySecret',
37      resave: false,
38      saveUninitialized: true,
39    })
40  );
41
```

Dark Mode: By clicking a button on any page, the application switches between dark mode and light mode.





Section 3: Application Technologies

The technologies used in our application include:

Nodejs

ExpressJs

EJS

JavaScript

Html

CSS

MongoDB

NodeJS – MongoDB : were used for storing information in the application, such as user accounts, campaign data, event information, and ticket storage. They were utilized in various pages of the application, including those mentioned above.

(Used in /About.html, /sign_up, /log_in, /update_profile, /events.html, /event-info.html, /buy_tickets.html, /updateUsers.html, /submit_seats, /updateUserProfile.html, /updateUserProfile, /deleteUser, /eventSearch.html, /getSearchQuery, /updateEvent.html, /updateEventInfo.html, /updateEventInfo,

/deleteEvent, /createEvent, /updateEvent, /updateReservation.html, /deleteReservation, /updateReservationInfo.html, /updateReservationInfo, /updateAbout.html, /updateAboutInfo)

HTML/CSS They were used for the aesthetic appearance of the page. All pages are written in HTML and call CSS files.

(Used in /About.html, events.html, /event-info.html, /buy_tickets.html, /updateUsers.html, /updateUserProfile.html, /updateEvent.html, /updateEventInfo.html, /updateReservation.html, /updateReservationInfo.html, /updateAbout.html, /homeAdmin.html, /homeT.html, /profile.html, /eventSearch)

Javascript/EJS: They were used for the interactivity of the user with the page. They change the displayed data based on user choices (e.g., Dark Mode, Search Bar) and show information retrieved from the database. All pages used EJS/JavaScript to update their information.

(Used in /About.html, events.html, /event-info.html, /buy_tickets.html, /updateUsers.html, /updateUserProfile.html, /updateEvent.html, /updateEventInfo.html, /updateReservation.html, /updateReservationInfo.html, /updateAbout.html, /homeAdmin.html, /homeT.html, /profile.html, /eventSearch)

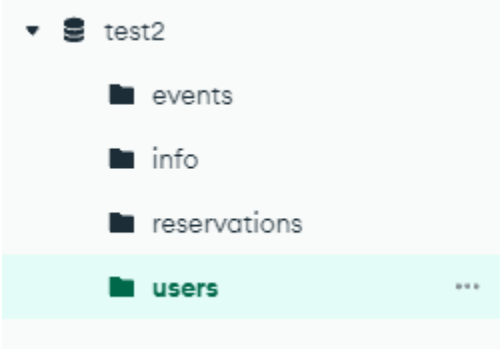
ExpressJs: It is the main component for page rendering and routing. It displays/retrieves information from the page based on user actions, such as form submission or opening a link.

All pages used Express.js for rendering

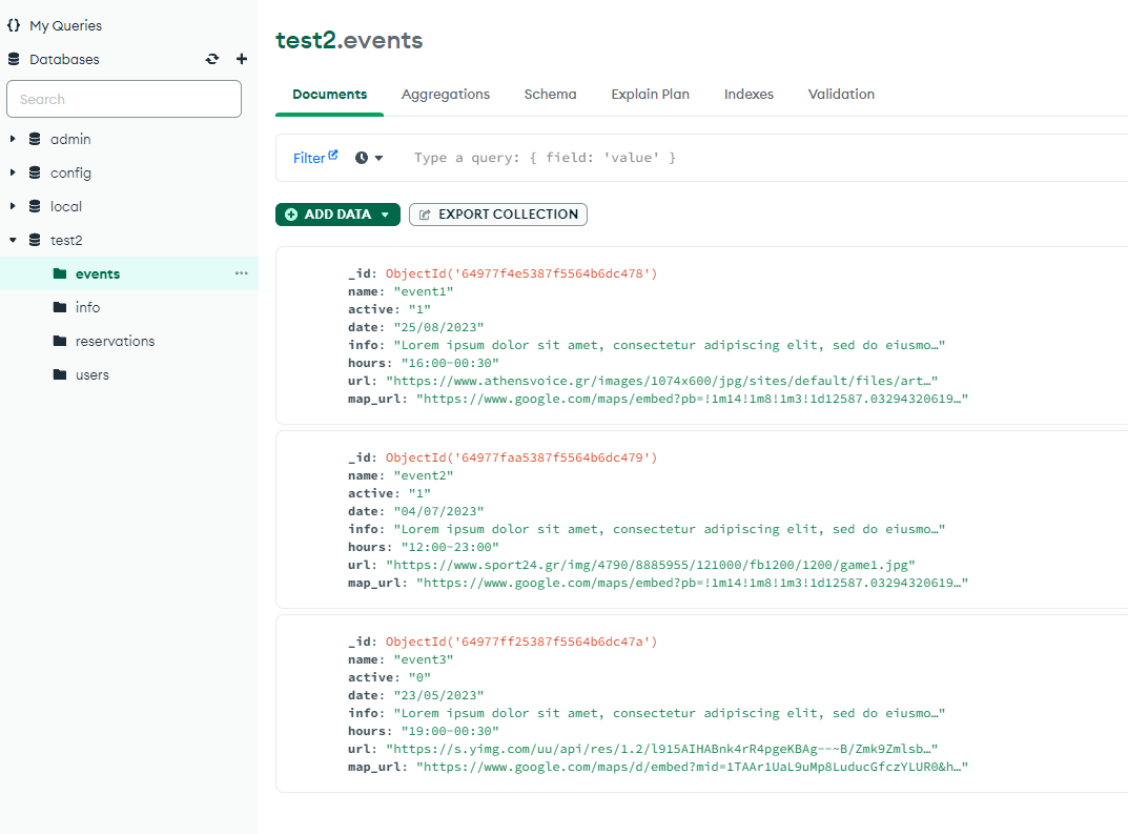
Section 4: Application User Manual

Database Analysis:

The created database is named "test2" and consists of 4 collections: "events," "info," "reservations," and "users."

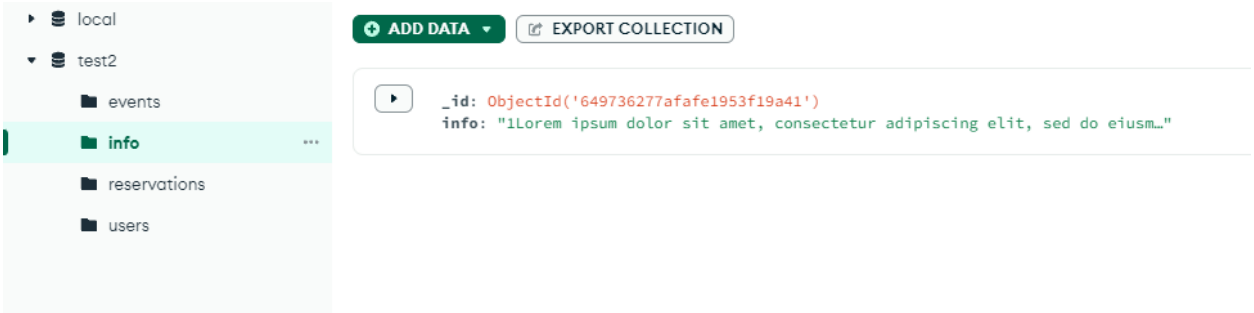


Firstly, the "events" collection contains documents related to events. Each document consists of the following fields:



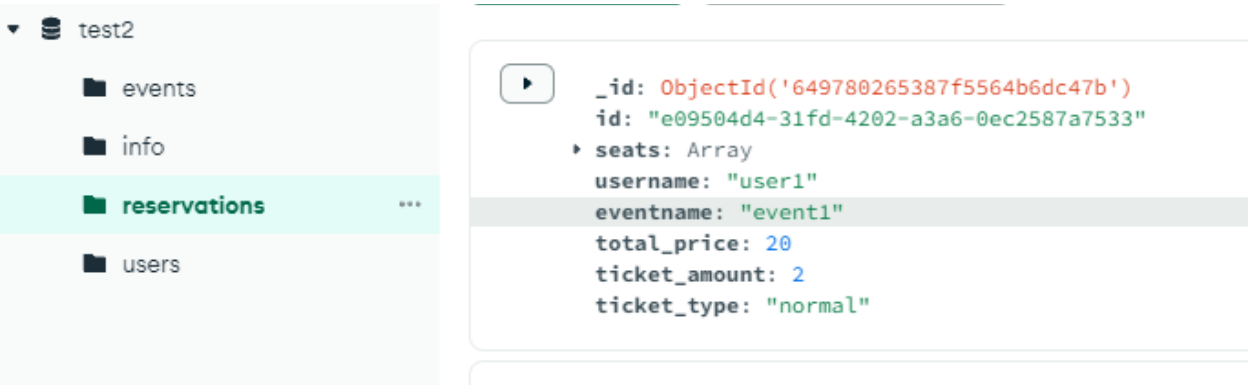
1. name: stores the name of the event
2. active: either 0 or 1, indicating whether the event is future (1) or not (0)
3. date: stores the date of the event
4. info: contains relevant information
5. hours: specifies the event hours
6. url: stores the URL of the event image on the information page
7. map_url: stores the URL of the event's location on Google Map

The "info" collection consists of documents with the following field:



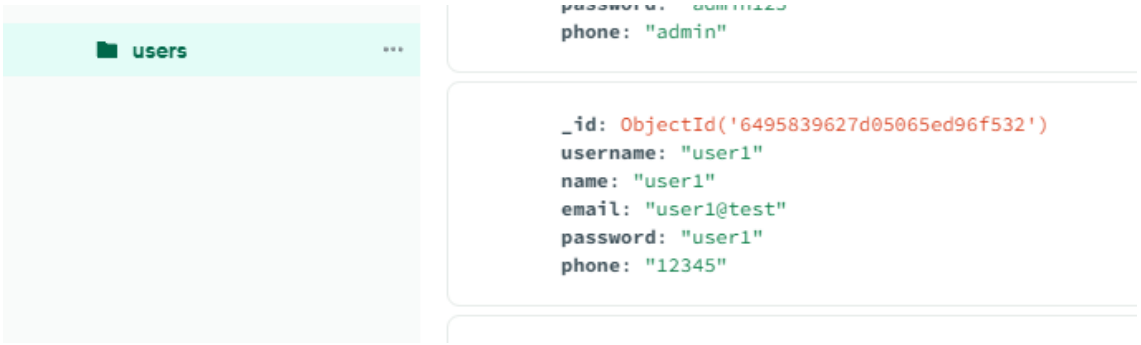
1. info: stores general campaign information

The "reservations" collection consists of documents with the following fields:



1. id: a unique ID for each reservation
2. seats: an array storing the number of seats selected by the user when purchasing tickets
3. username: the username of the user who made the reservation
4. event name: the name of the event for which the reservation was made
5. total_price: the total price to be paid
6. ticket_amount: the number of tickets purchased
7. ticket_type: the type of tickets purchased by the user

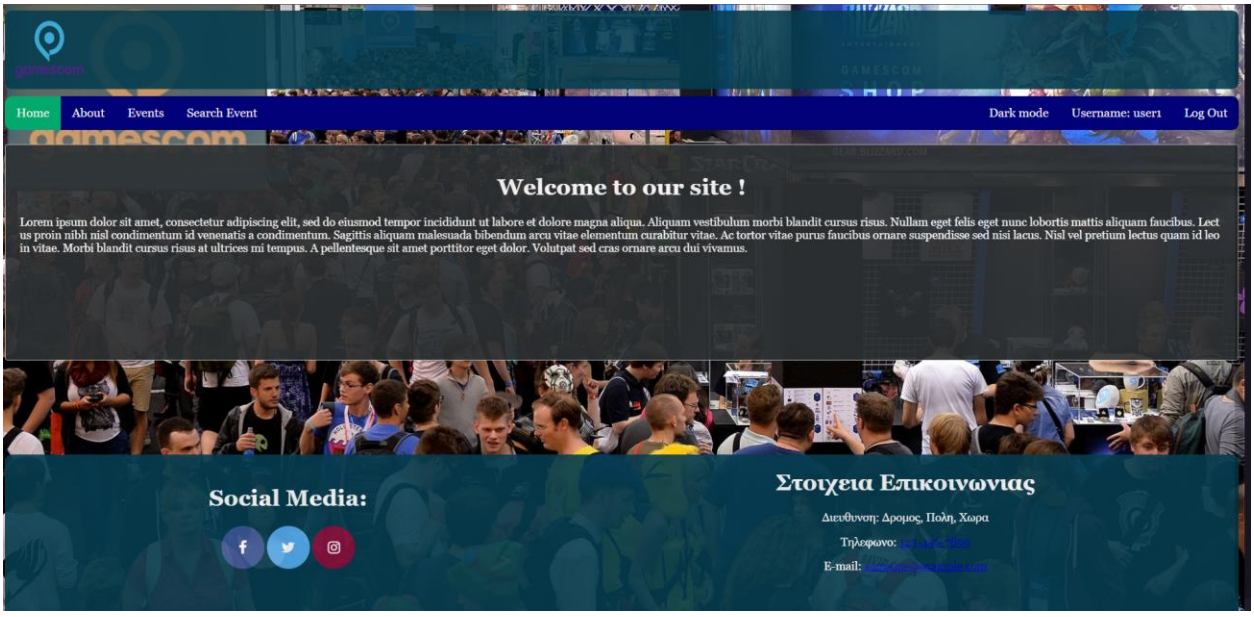
The "users" collection consists of documents with the following fields:



1. username: a unique username automatically assigned during user registration
2. name: the user's name (used for creating the username)
3. email: the user's email
4. password: the user's password
5. phone: the user's phone number

Media Queries Analysis:

The media queries were placed in the "styles.css" file in the "other" folder. Initially, on a desktop, the homepage appears as follows:



The first media query used targets devices with a screen width between 472 and 768 pixels.

```
/* Specific styles for phone devices up to 768px */
@media (max-width: 768px) and (min-width: 472px) {
    .topnav a:not(:first-child), .dropdown .dropbtn {
      display: none;
    }
    .topnav a.icon {
      float: right;
      display: block;
    }

    .topnav.responsive {position: relative;}
    .topnav.responsive a.icon {
      position: absolute;
      right: 0;
      top: 0;
    }
    #signup{
      float: left;
  }
    .topnav.responsive a {
      float: none;
      display: block;
      text-align: left;
    }
    .topnav.responsive .dropdown {float: none;}
    .topnav.responsive .dropdown-content {position: relative;}
    .topnav.responsive .dropdown .dropbtn {
      display: block;
      width: 100%;
      text-align: left;
    }

    header{
      padding: 0px;
      font-size: 24px;
      height: 100px;
    }

    .info-box{
      font-size: 15px;
    }

    footer{
      max-height: 150px;
      padding: 0px;

    }
```

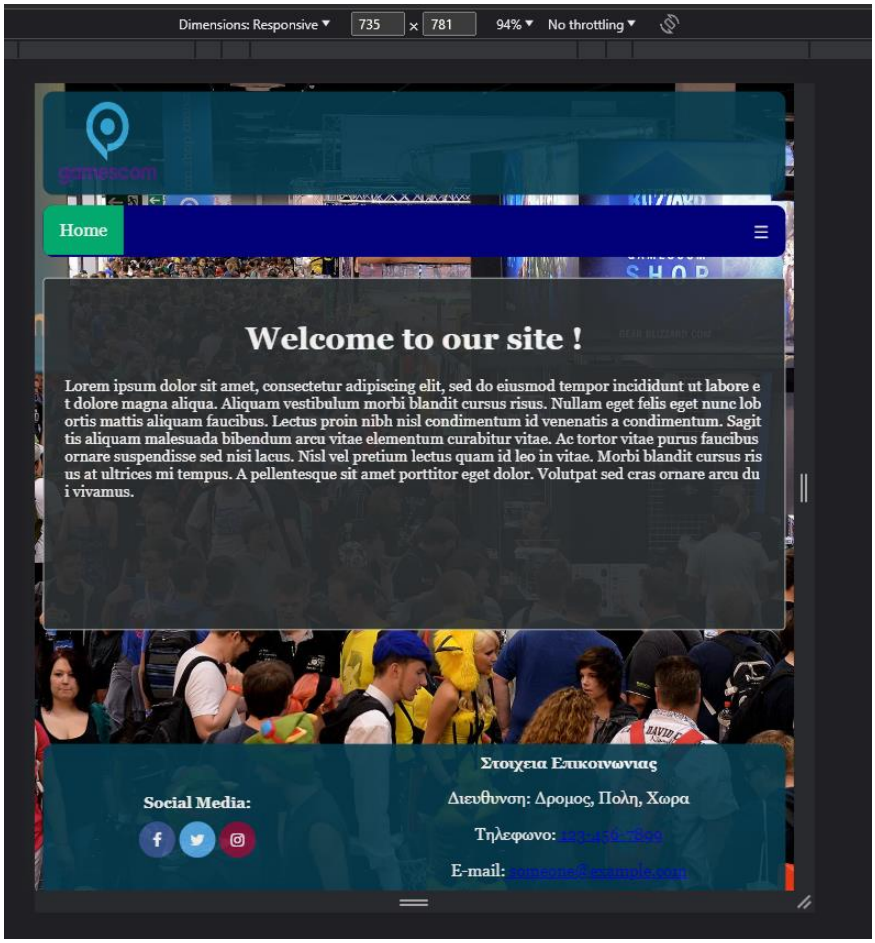
```
    }

    .fa{
      width: 35px;
      padding: 10px;
      font-size: 15px;
    }

    footer>div>div>h1{
      font-size: 15px;
    }
    .child>img{
      width: 250px;
    }
}
```

The first child of the topnav bar will remain visible while all others will become options in the dropdown menu. It is also positioned on the left and the "responsive" class is applied to the "topnav" class. Additionally, the header and the "info-box" that contains basic body information presented to the user undergo some changes. Furthermore, the size of the footer and the font size changes. Lastly, the size of the logo image on the header is modified.

Thus, the initial page will look something like this when viewed on devices with these specific dimensions:

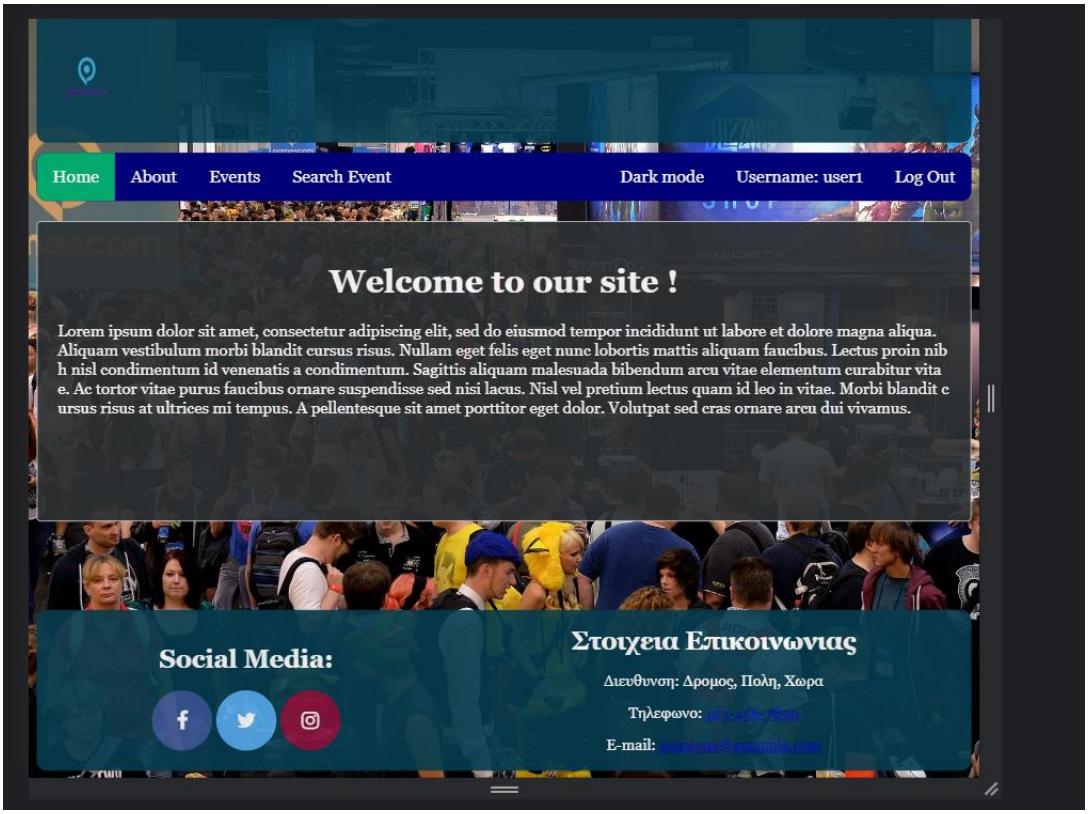


The next media query created is for devices between 992 and 768 pixels. These changes mainly apply to tablets and devices with relatively large screens, so not many things are altered. The modifications include the size of the header, footer, and the logo image.

```css
/* Specific styles for tablet devices */
@media (max-width: 992px) and (min-width:768px) {

  header{
    padding: 0px;
    font-size: 24px;
    height: 130px;
  }
  header>img{
    width: 100px;
    height:100px;
    align-self: center;
    padding: 30px;
  }

  footer{
    height: 160px;
    padding: 0px;
    font-size: 15px;
  }
  footer>div>div>h1{
    font-size: 25px;
  }
  .child>img{
    width: 400px;
  }
}
```



Finally, we have the media query for mobile devices and devices with very small screens.

```css
@media (max-width: 472px) {
  .topnav a:not(:first-child), .dropdown .dropbtn {
    display: none;
  }
  .topnav a.icon {
    float: right;
    display: block;
  }

  .topnav.responsive {position: relative;}
  .topnav.responsive a.icon {
    position: absolute;
    right: 0;
    top: 0;
  }
  #signup{
    float: left;
}
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
  .topnav.responsive .dropdown {float: none;}
  .topnav.responsive .dropdown-content {position: relative;}
  .topnav.responsive .dropdown .dropbtn {
    display: block;
    width: 100%;
    text-align: left;
  }

  .fa{
    width: 35px;
    padding: 10px;
    font-size: 15px;
  }

 footer{
  font-size: 10px;
  max-height: 150px;
 }
 .child>img{
  width: 100px;
}
}
```

Here, the changes are similar to the changes made in the first media query for devices between 472 and 768 pixels. However, here the elements on the screen shrink even more to fit nicely on the screen.

Installation Guide for the Application:

The web application uses MongoDB for the database, NodeJS with ExpressJS for the backend, and HTML with EJS for the frontend.

System Requirements:

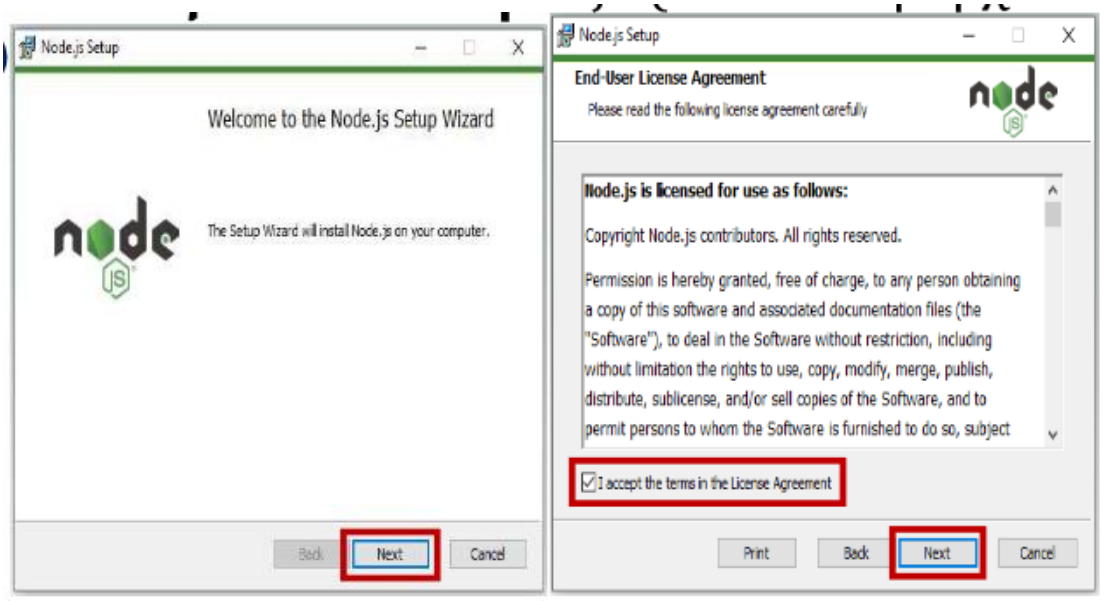
Operating System: Windows

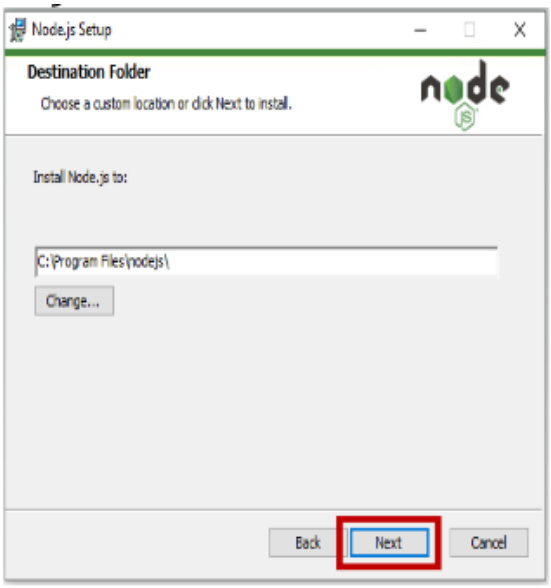
MongoDB: Version 6.0.6 or later

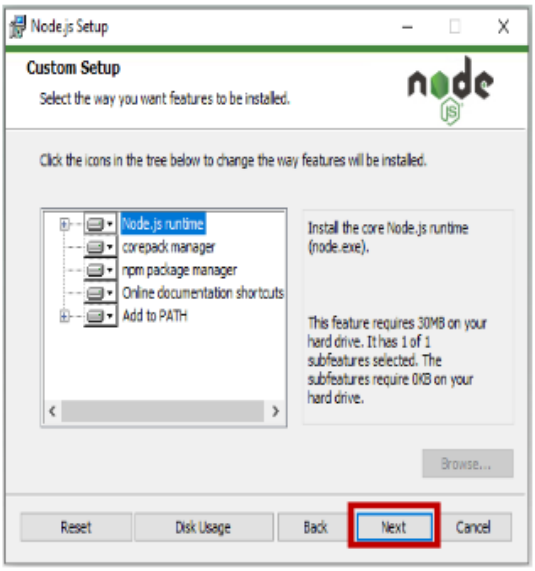
NodeJS: Version 18.16.0 or later

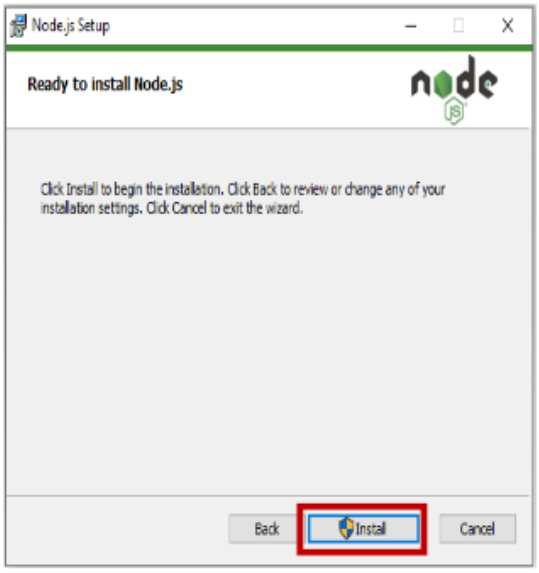
Step 1: Installing NodeJS

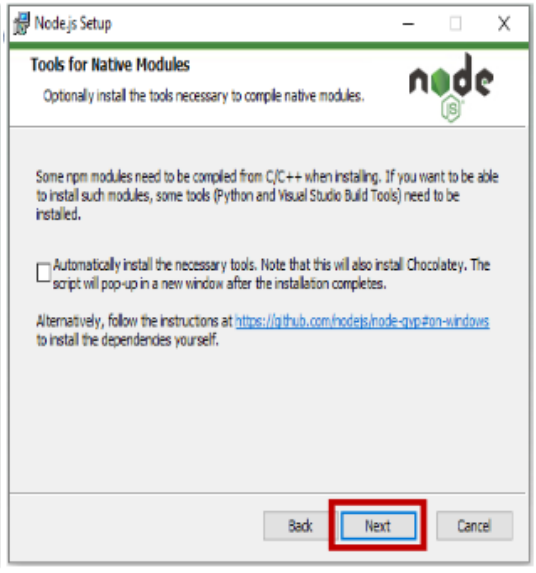
1) Visit the NodeJS website at https://nodejs.org/en/#download and install the recommended version.
2) Follow the steps shown in the images for the proper installation of NodeJS.

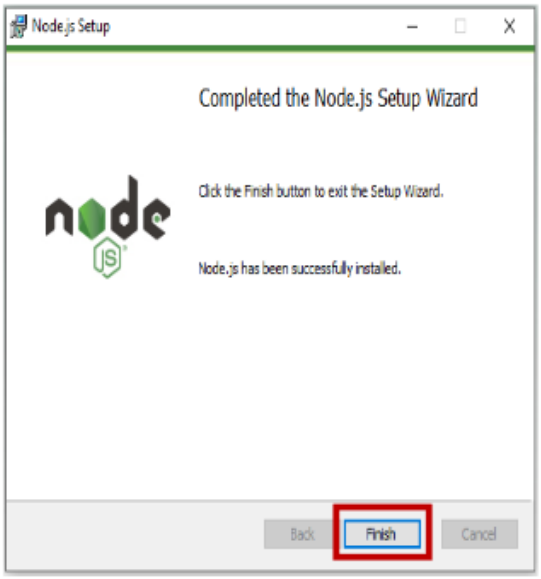



Node.js Setup
Destination Folder
Choose a custom location or click Next to install.
Install Node.js to:
C:\Program Files\nodejs\
Change...
Back
Next
Cancel


Node.js Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Node.js runtime
corepack manager
npm package manager
Online documentation shortcuts
Add to PATH
Install the core Node.js runtime (node.exe).
This feature requires 30MB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 0KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel


Node.js Setup
Ready to install Node.js
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel


Node.js Setup
Tools for Native Modules
Optionally install the tools necessary to compile native modules.
Some npm modules need to be compiled from C/C++ when installing. If you want to be able to install such modules, some tools (Python and Visual Studio Build Tools) need to be installed.
Automatically install the necessary tools. Note that this will also install Chocolatey. The script will pop-up in a new window after the installation completes.
Alternatively, follow the instructions at https://github.com/nodejs/node-gyp#on-windows to install the dependencies yourself.
Back
Next
Cancel

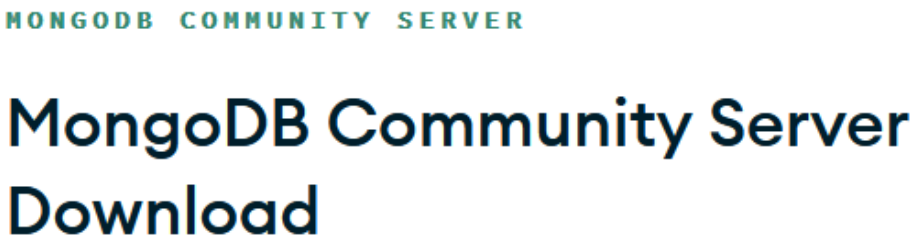


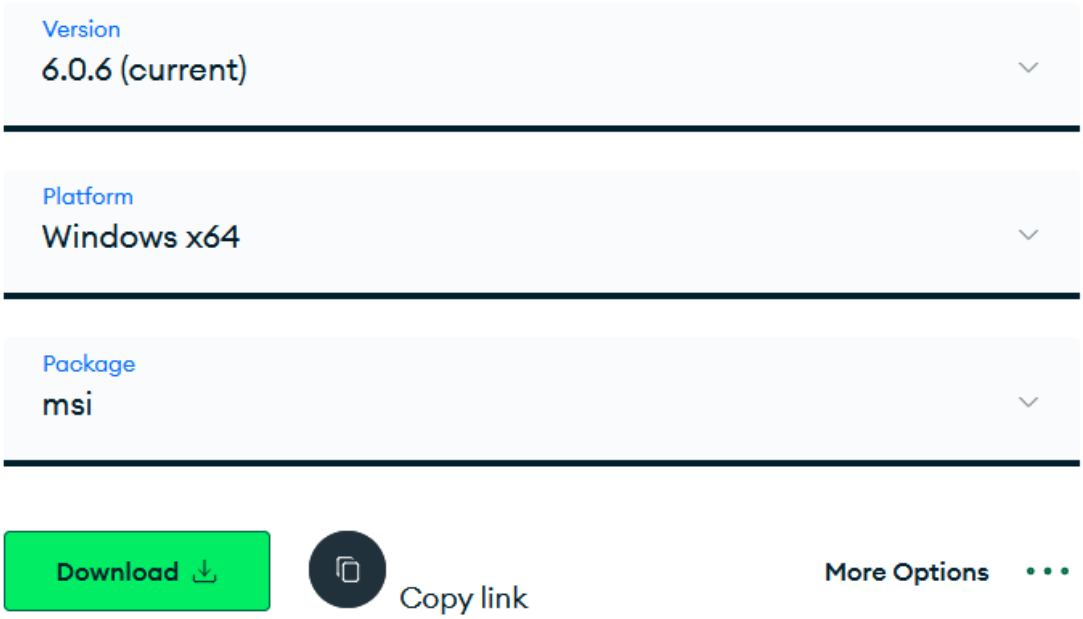
3) After restarting, we check if the installation was successful.

```
C:\Users\User\Desktop\dprog\ergasiaDProgrammatismos>node -v
v18.16.0
```

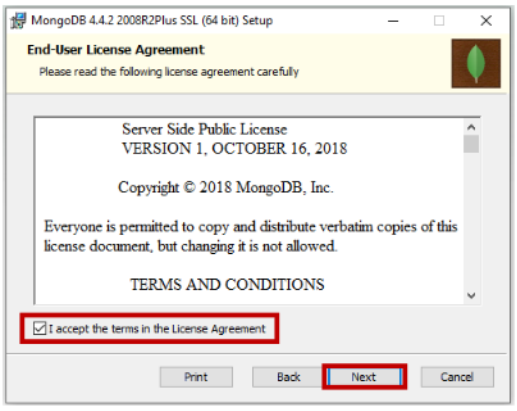
Step 2: Installing MongoDB

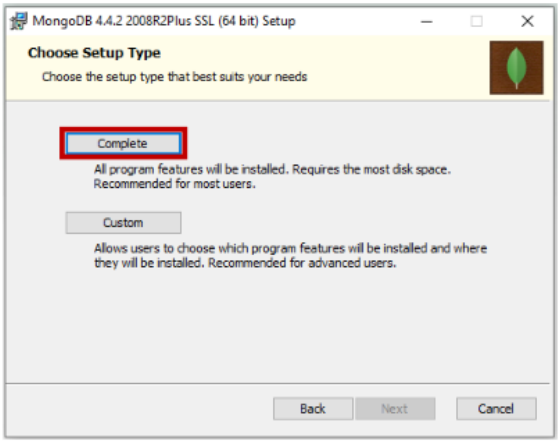
1) Visit the MongoDB website and install MongoDB Community Server: https://www.mongodb.com/try/download/community

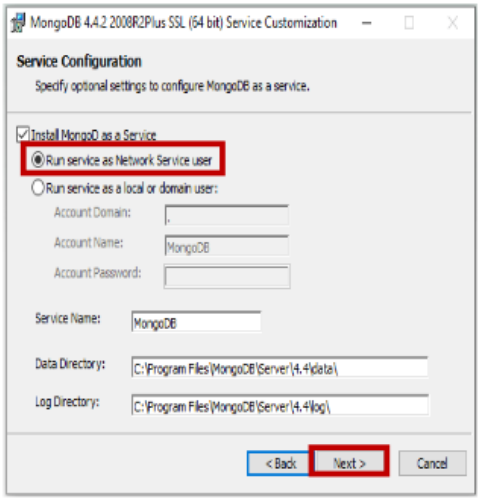


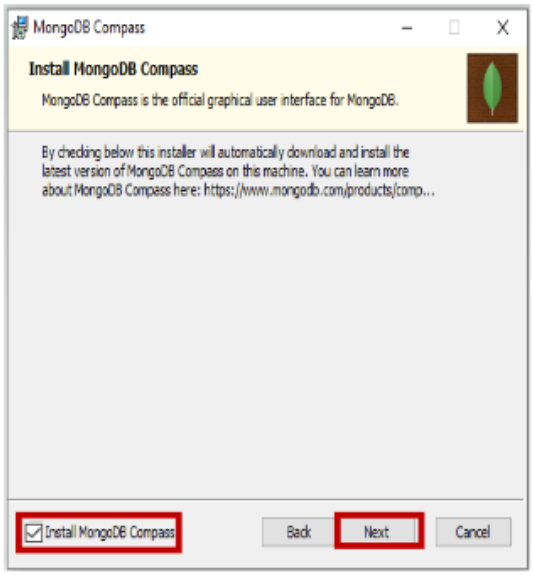


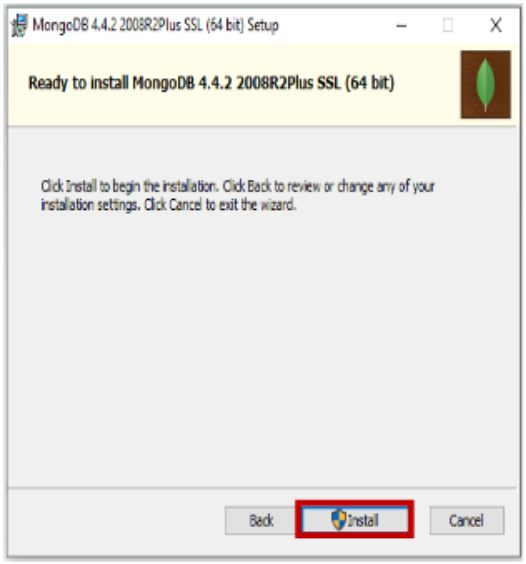
2) Follow the steps shown in the images for the proper installation.





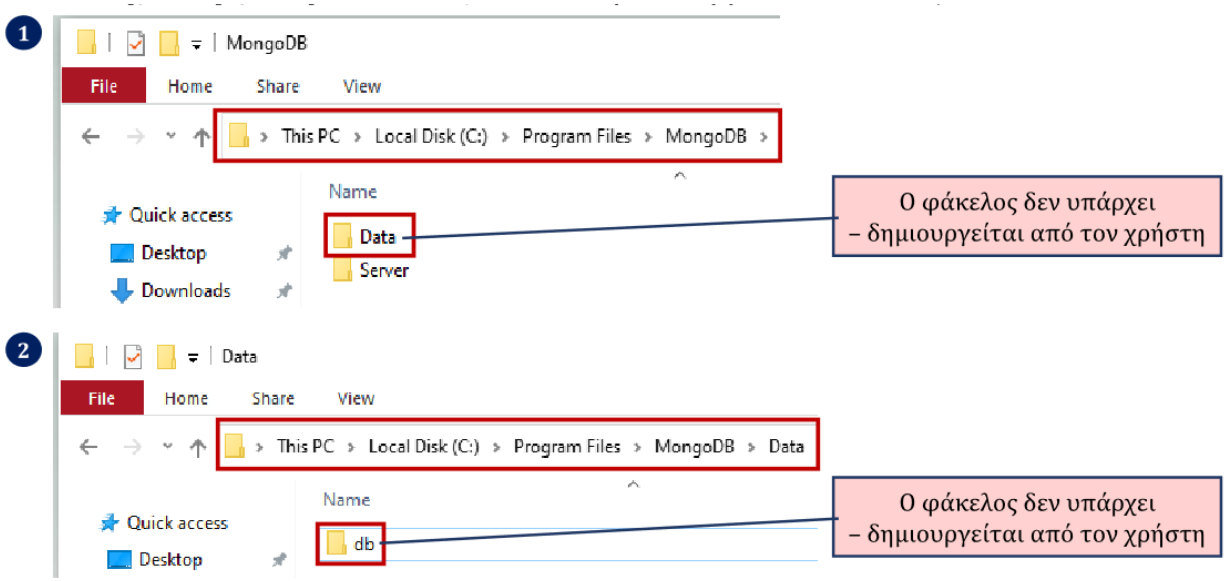



MongoDB Compass
Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/comp...
Install MongoDB Compass
Back
Next
Cancel


MongoDB 4.4.2 2008R2Plus SSL (64 bit) Setup
Ready to install MongoDB 4.4.2 2008R2Plus SSL (64 bit)
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

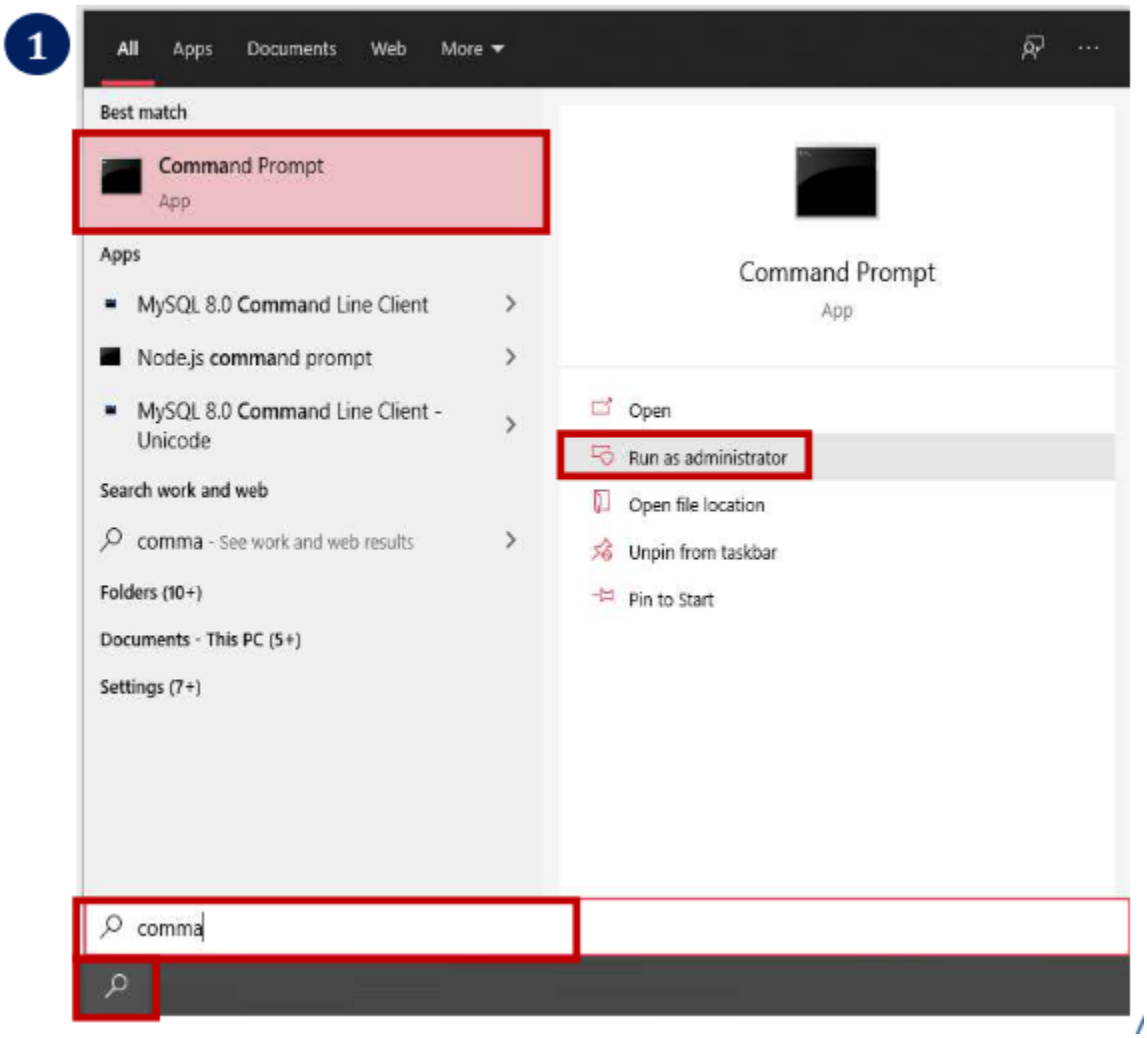

MongoDB 4.4.2 2008R2Plus SSL (64 bit) Setup
Completed the MongoDB 4.4.2 2008R2Plus SSL (64 bit) Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

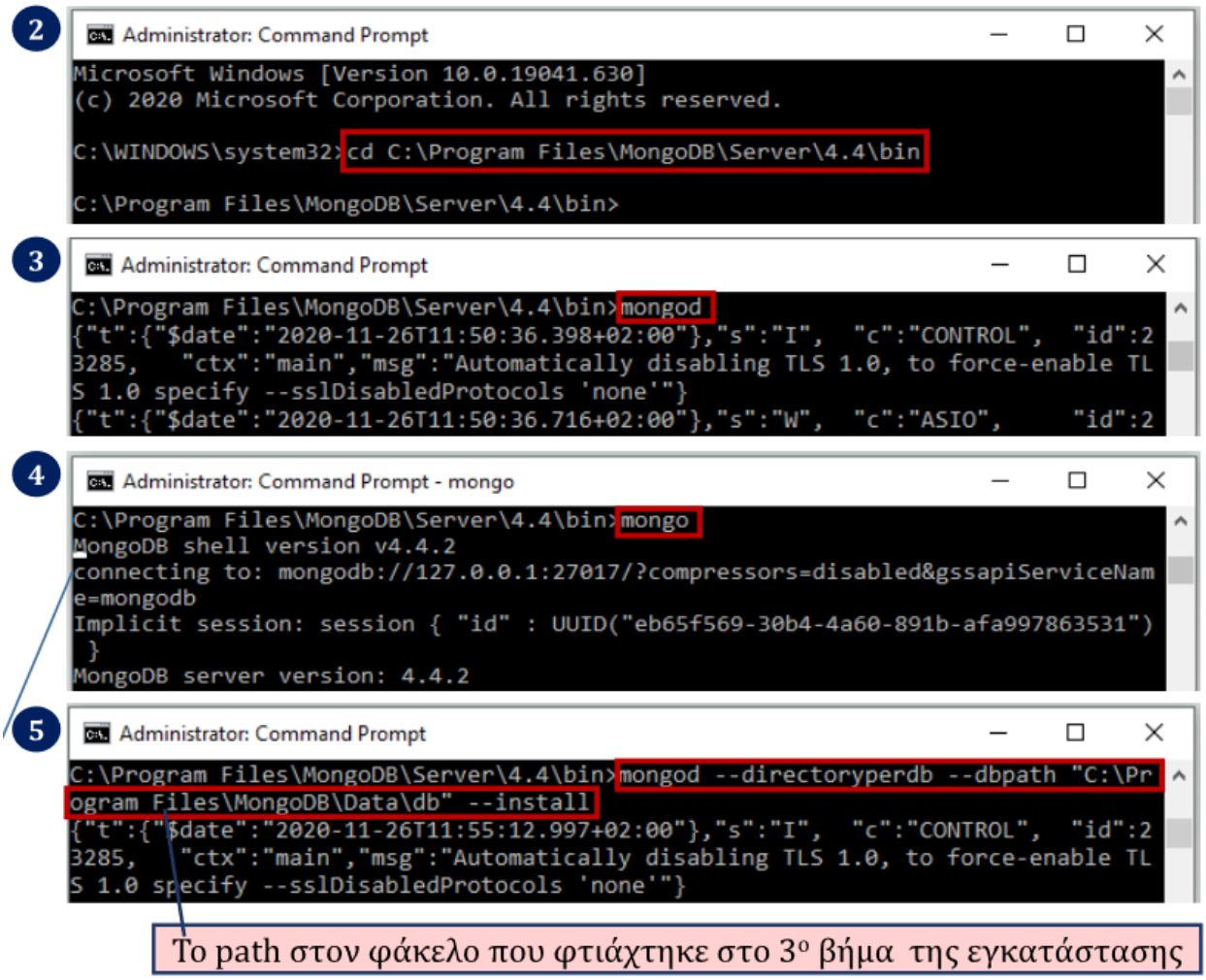
3) To ensure the smooth startup of the MongoDB Server and the functioning of the database, we create a folder to store the databases.



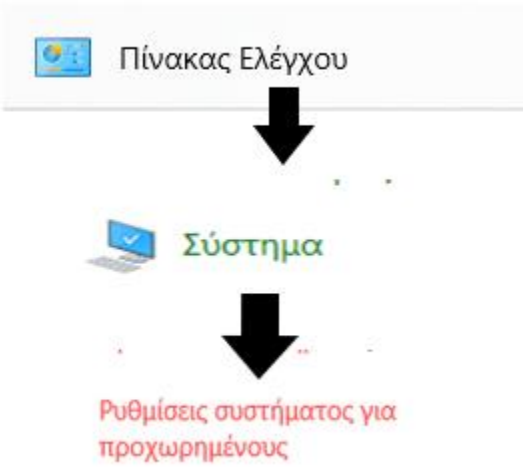
(**Note: at the beginning of the project, we encountered an issue with starting the MongoDB server, and we resolved it using the method found in the MongoDB forums.**)

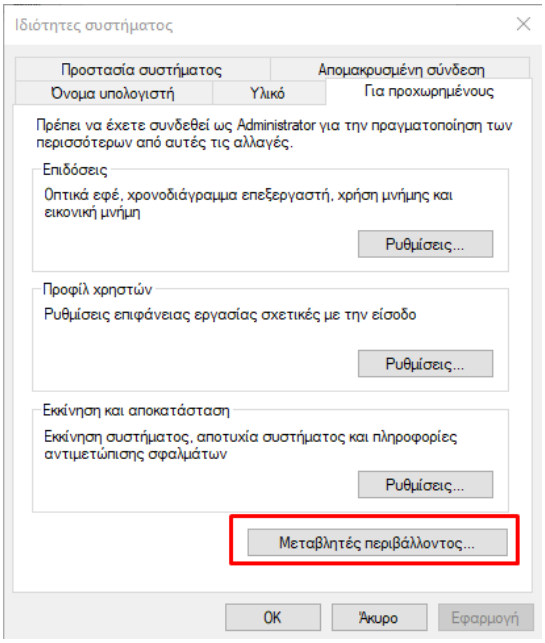
4) MongoDB Configuration

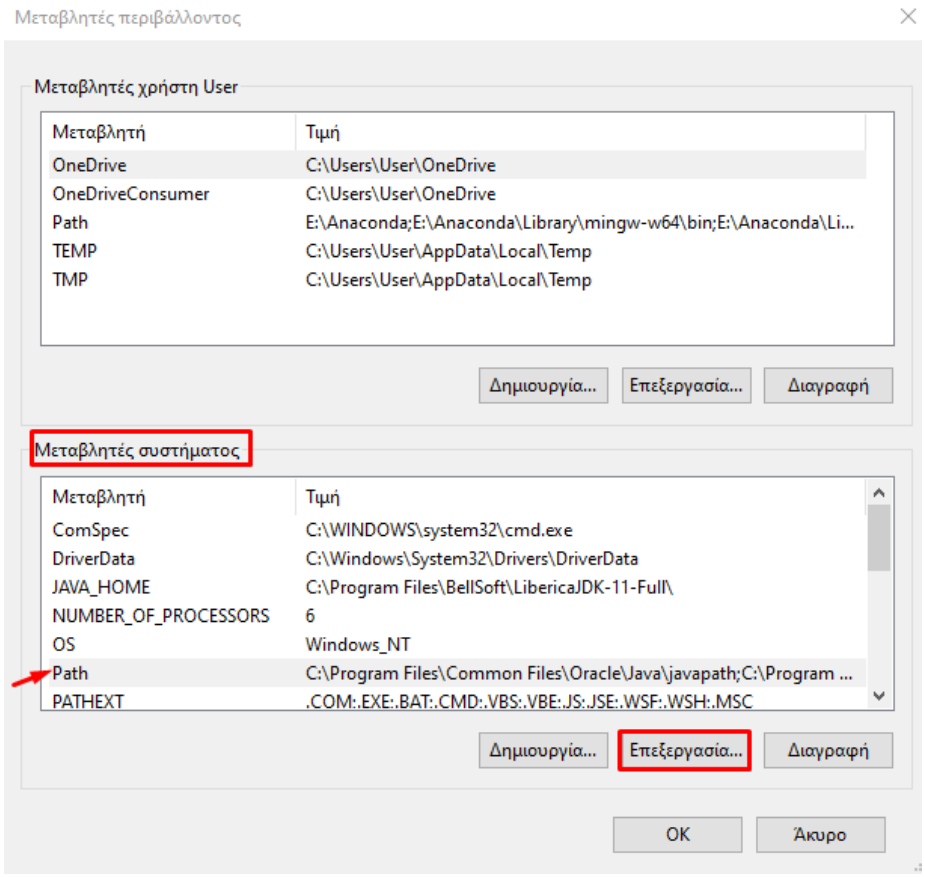


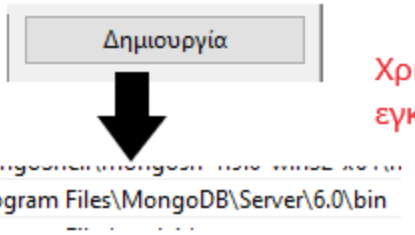


5) Update the system variable Path.

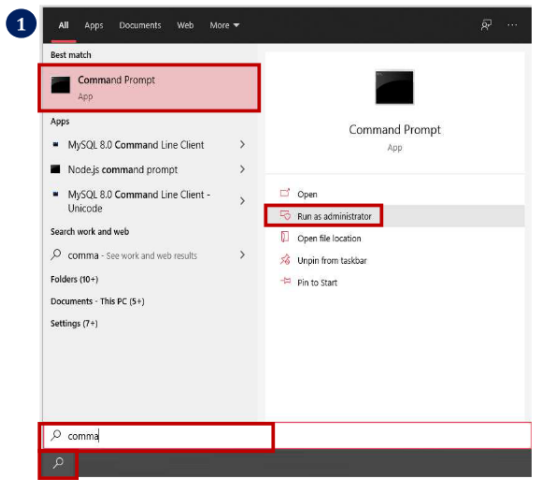

Πίνακας Ελέγχου
Σύστημα
Ρυθμίσεις συστήματος για προχωρημένους

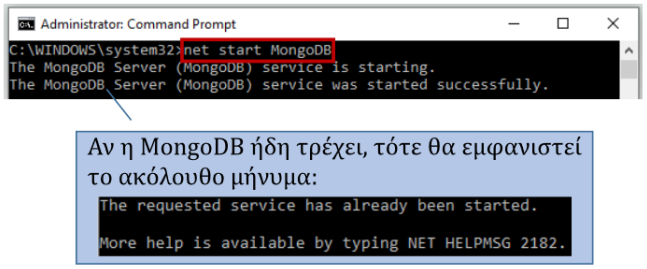

Ιδιότητες συστήματος
Προστασία συστήματος
Απομακρυσμένη σύνδεση
Όνομα υπολογιστή
Υλικό
Για προχωρημένους
Πρέπει να έχετε συνδεθεί ως Administrator για την πραγματοποίηση των περισσότερων από αυτές τις αλλαγές.
Επιδόσεις
Οπτικά εφέ, χρονοδιάγραμμα επεξεργαστή, χρήση μνήμης και εικονική μνήμη
Ρυθμίσεις...
Προφίλ χρηστών
Ρυθμίσεις επιφάνειας εργασίας σχετικές με την είσοδο
Ρυθμίσεις...
Εκκίνηση και αποκατάσταση
Εκκίνηση συστήματος, αποτυχία συστήματος και πληροφορίες αντιμετώπισης σφαλμάτων
Ρυθμίσεις...
Μεταβλητές περιβάλλοντος...
OK
Άκυρο
Εφαρμογή


Μεταβλητές περιβάλλοντος
Μεταβλητές χρήστη User
Μεταβλητή
Τιμή
OneDrive
C:\Users\User\OneDrive
OneDriveConsumer
C:\Users\User\OneDrive
Path
E:\Anaconda;E:\Anaconda\Library\mingw-w64\bin;E:\Anaconda\Li...
TEMP
C:\Users\User\AppData\Local\Temp
TMP
C:\Users\User\AppData\Local\Temp
Δημιουργία...
Επεξεργασία...
Διαγραφή
Μεταβλητές συστήματος
Μεταβλητή
Τιμή
ComSpec
C:\WINDOWS\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
JAVA_HOME
C:\Program Files\BellSoft\LibericaJDK-11-Full\
NUMBER_OF_PROCESSORS
6
OS
Windows_NT
Path
C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program ...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
Δημιουργία...
Επεξεργασία...
Διαγραφή
OK
Άκυρο



6) Start MongoDB.





- **Τερματισμός** MongoDB:

7) Start Mongo Shell or Install Mongo Compass and start it to view the databases.
8) Check MongoDB.

```
mongo --version
```

9) Install the MongoDB Node module via the command prompt (cmd) by following the steps on the website:https://www.npmjs.com/package/mongodb

```
npm init -y
```

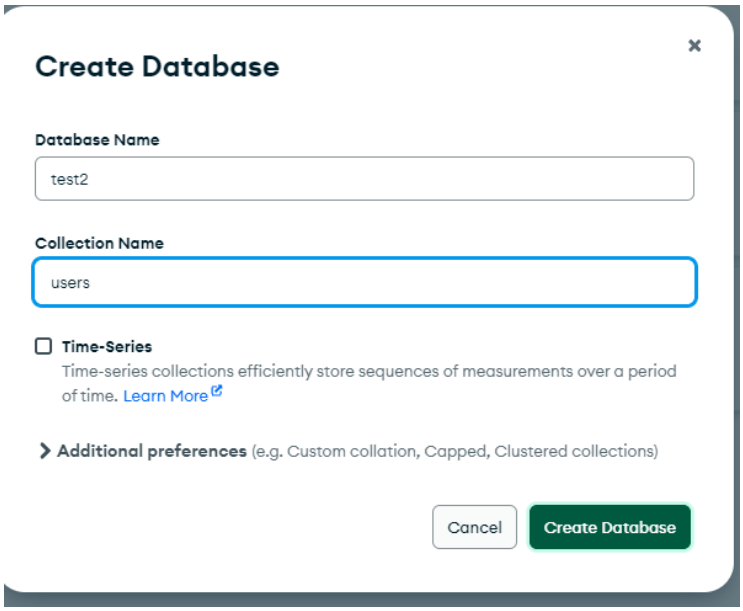
Next, install the driver as a dependency.

```
npm install mongodb
```

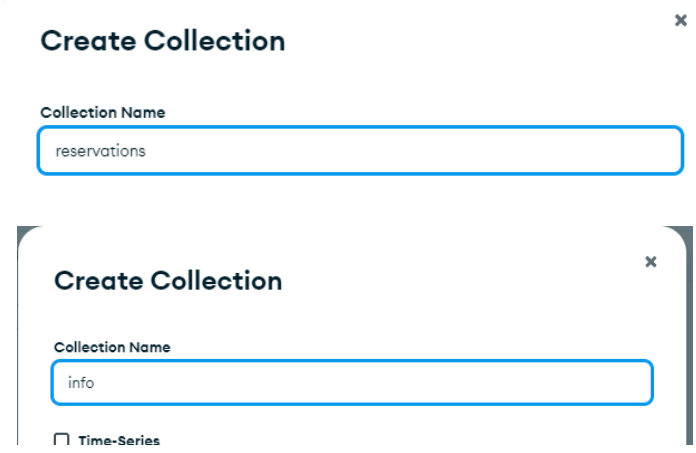
(**Note**: the 'npm init -y' command creates a package.json file with application details. The application does not require it as the folder already contains it.)

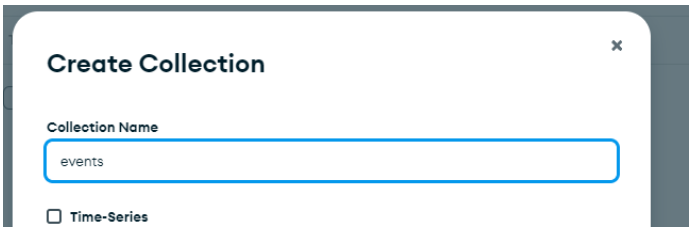
10) Creating MongoDB database and collections:

To create the required database with the specific name, we need to create a database named "test2" and an initial collection named "users".

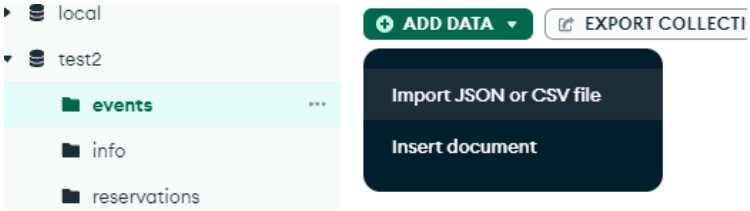


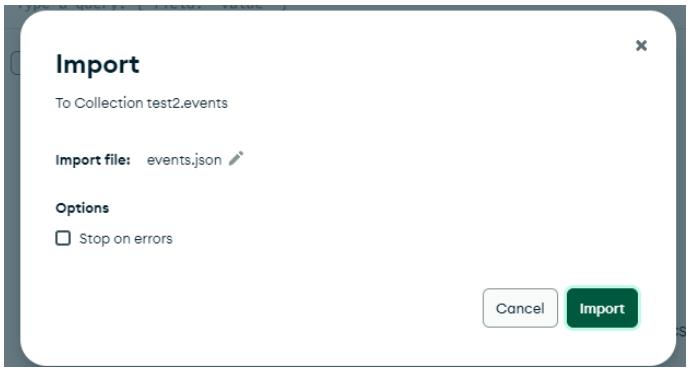
Next, we create the remaining collections that correspond to the JSON files that will be imported.

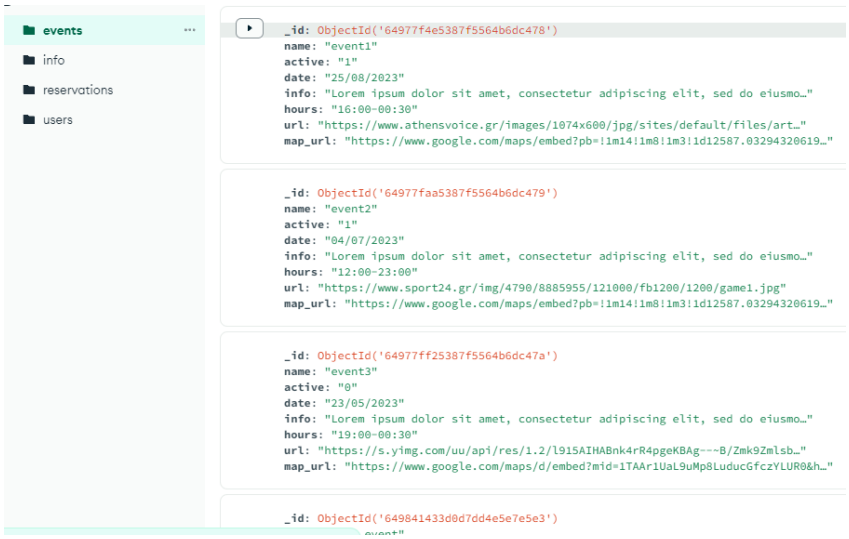




Finally, we navigate to each collection and import the respective JSON file.







The same process is repeated for the other collections.

Step 3: Unzip the application folder and run the command 'npm install' to update or install all dependencies from the node_modules folder.

```
npm install
```

Step 4: Open MongoDB to store logs.

```
C:\Users\User\Desktop\dprog\ergasiaDProgrammatismos>mongod
{"t":{"$date":"2023-06-25T15:42:32.871+03:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInterna
lClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient"
:true}}}
{"t":{"$date":"2023-06-25T15:42:32.872+03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1","msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2023-06-25T15:42:34.155+03:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"thread1","msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2023-06-25T15:42:34.157+03:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
{"t":{"$date":"2023-06-25T15:42:34.157+03:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationRecipientService","namespace":"config.tenantMigratio
```

Step 5: Execute the command to start the system and navigate to the 'localhost:3000' page to use the application.

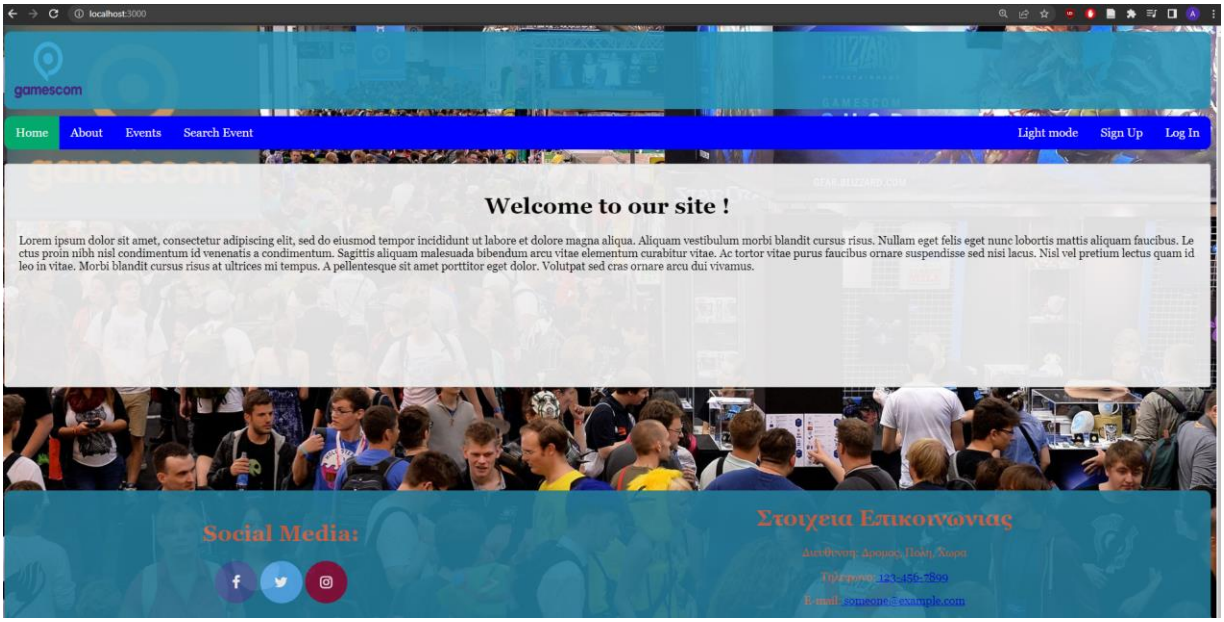
```
C:\Users\User\Desktop\dprog\ergasiaDProgrammatismos>node index.js
Server is listening on http://localhost:3000
```

Application Analysis:

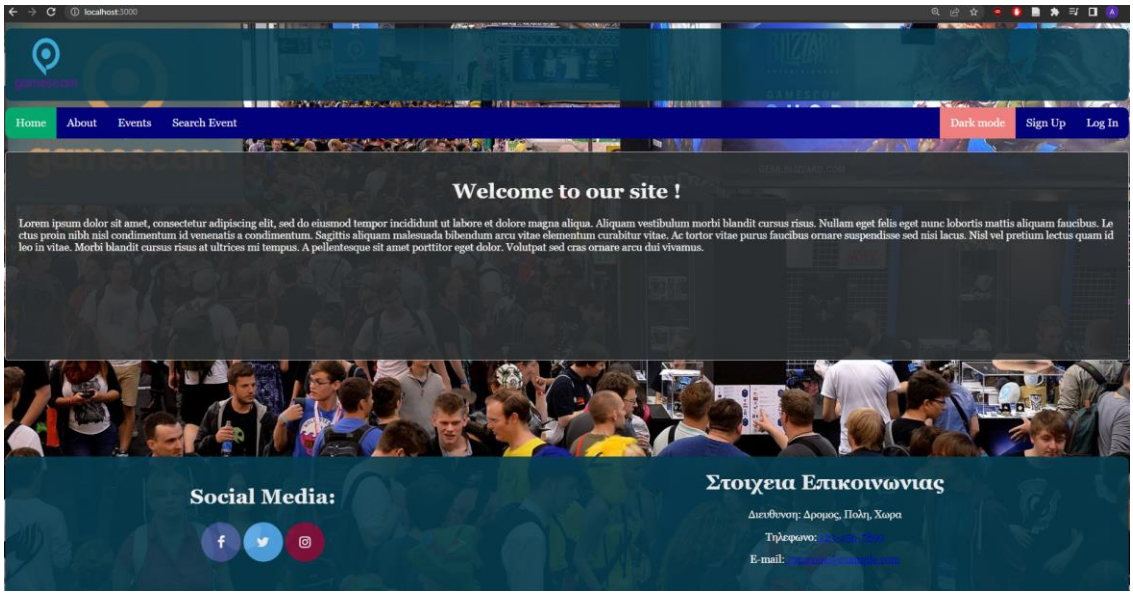
First, let's analyze the application from the user's perspective, and then from the perspective of the page administrator to present all the functionalities.

(1) Home Page:

When the user enters the website, they will initially land on the home page:

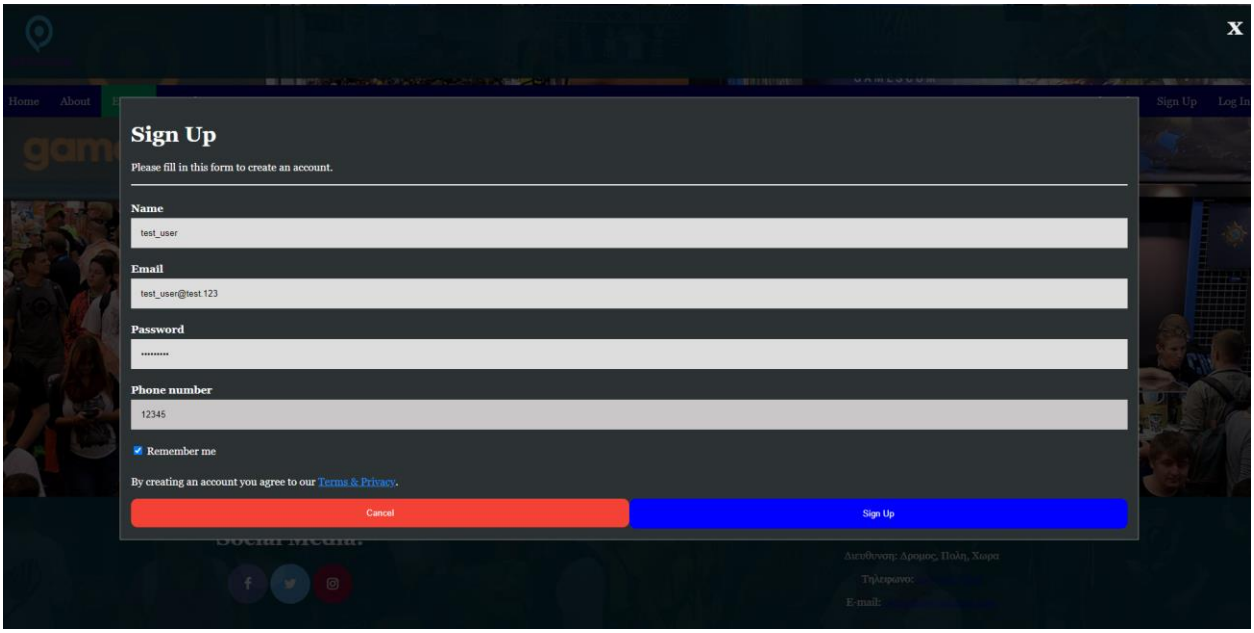


From there, they can navigate elsewhere by clicking on the options available in the navbar. They can also change the theme of the pages to dark or light mode and sign up or log in.

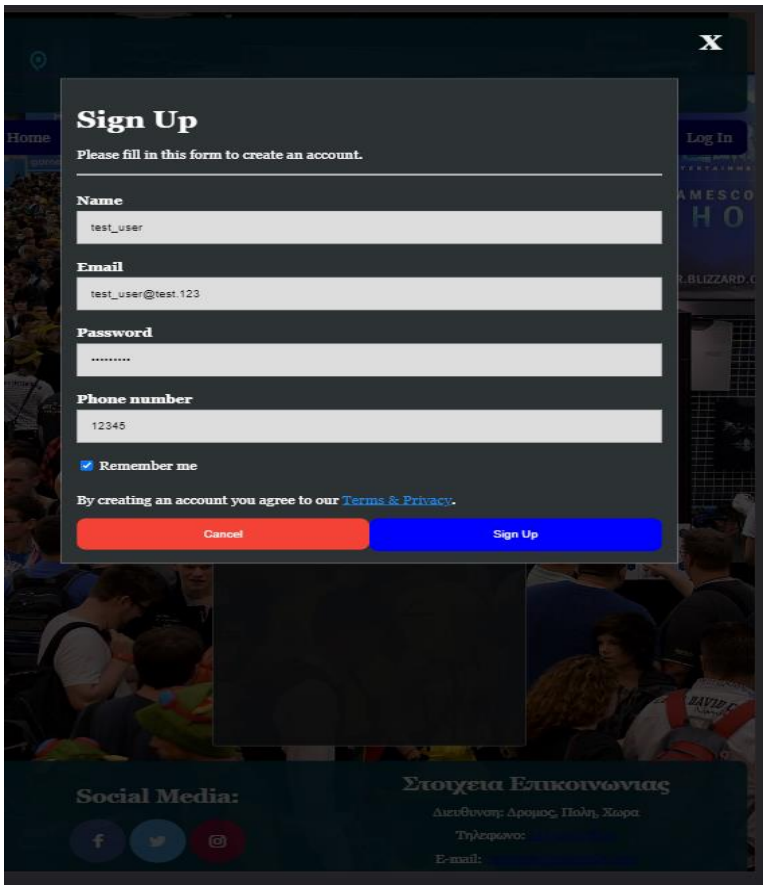


(2) User Registration:

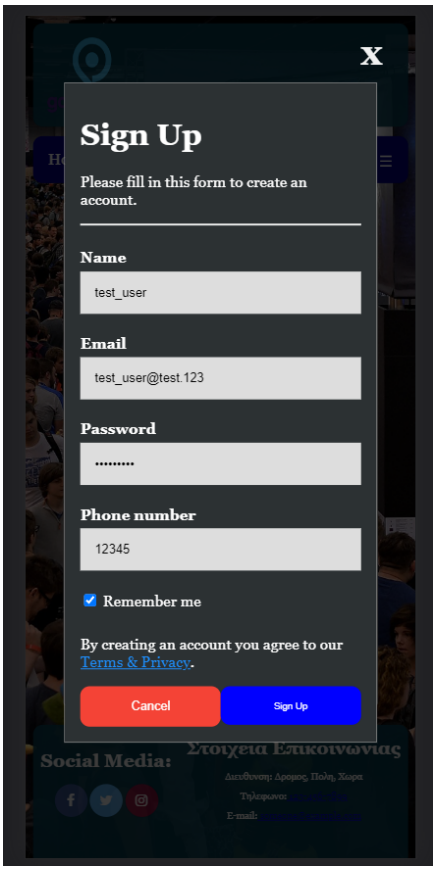
ow, let's look at the user registration process. From any page, the user can sign up by clicking on the "Sign Up" option in the top navigation bar. This will display a modal form with fields to be filled out, such as "name," "email," "password," and "phone number."



Sign up in Desktop



Tablet Sign up



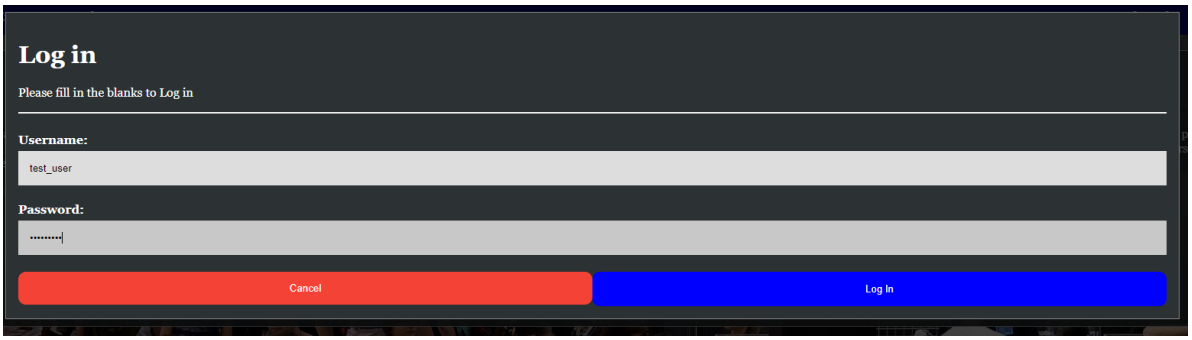
Smartphone Sign up

By clicking the 'x' or anywhere outside the form, the form will close. Clicking "Sign Up" will send the inputs to the database, creating a new document. After entering their desired name, a unique username is assigned to the user based on their name.

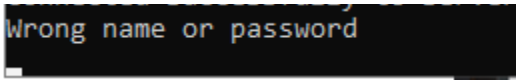

```
_id: ObjectId('649829a53d0d7dd4e5e7e5e0')
username: "test_user"
name: "test_user"
email: "test_user@test.123"
password: "test_user"
phone: "12345"
```

(3) User Login:

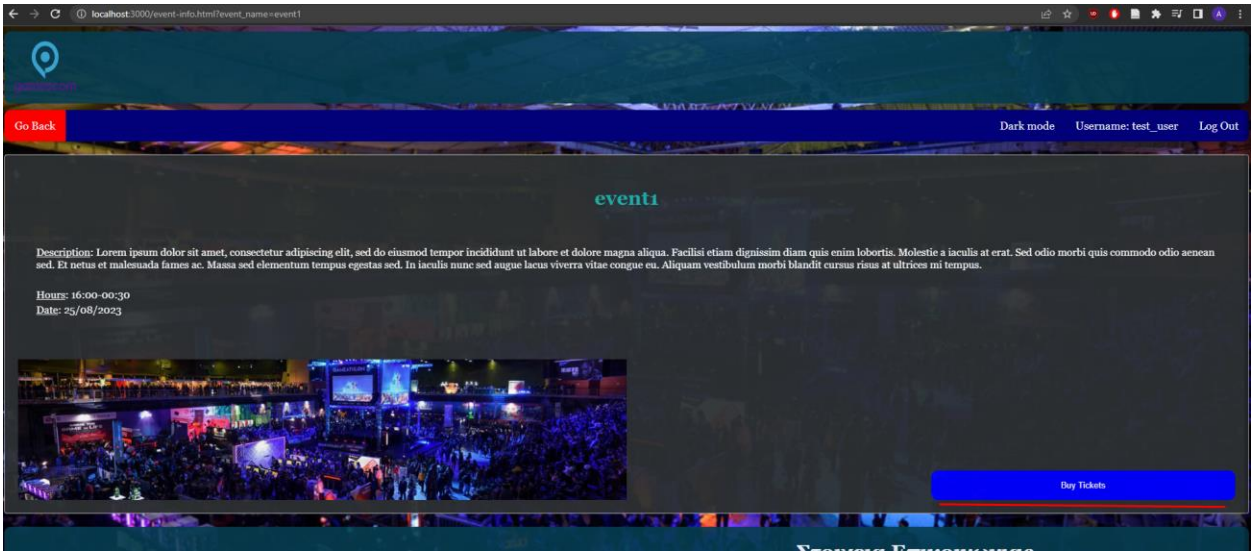
After signing up, the user can log in. When signing up, the user is automatically logged in, but in any other case, they will need to enter their username and password to log in again.



If either of the credentials is incorrect, the user will not be able to log in, and the following message will be displayed in the console.

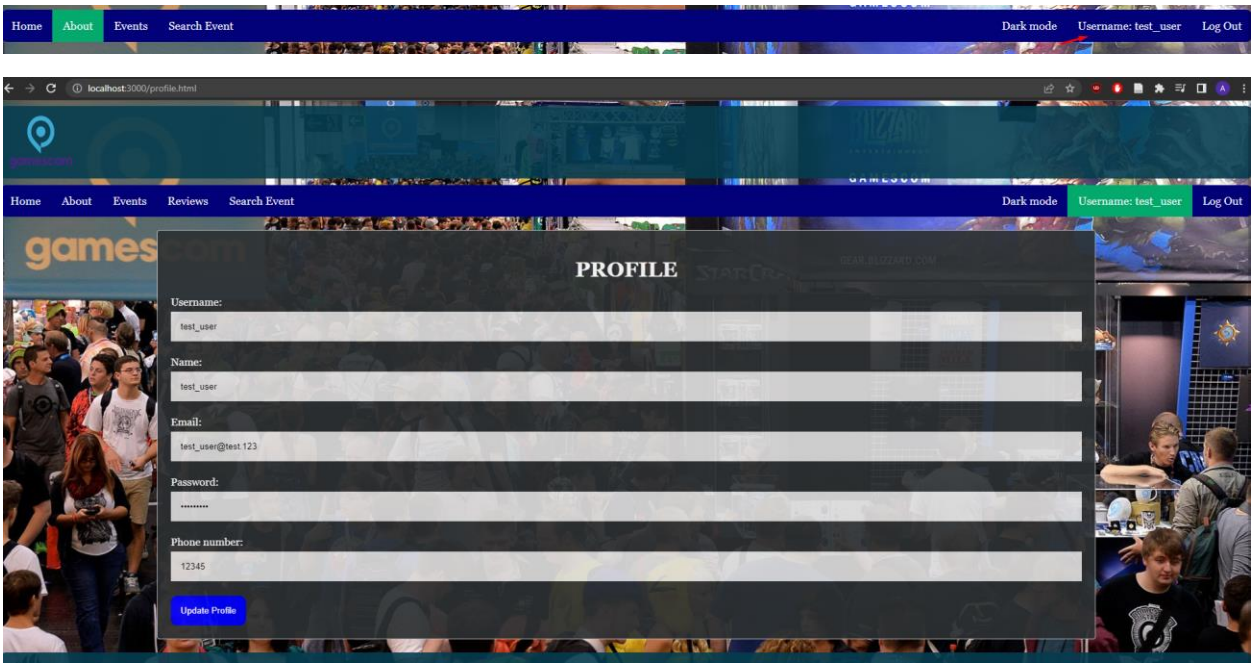


Once the user has successfully logged in, they can perform the same actions as before and also click on the "Buy Tickets" button that will appear on the page with information about a specific event to purchase tickets for it.

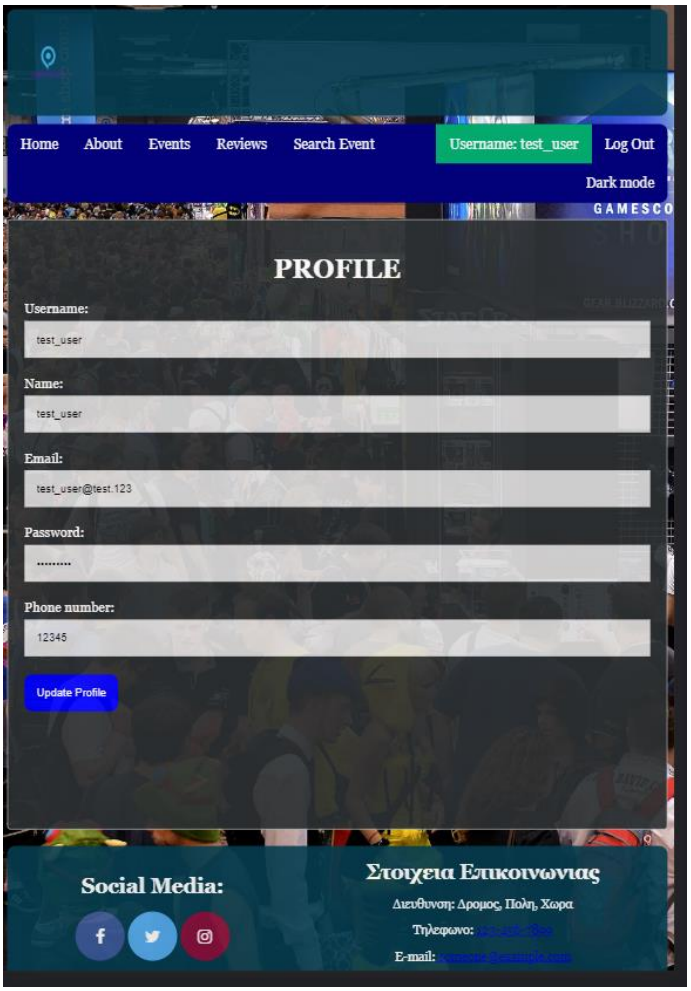


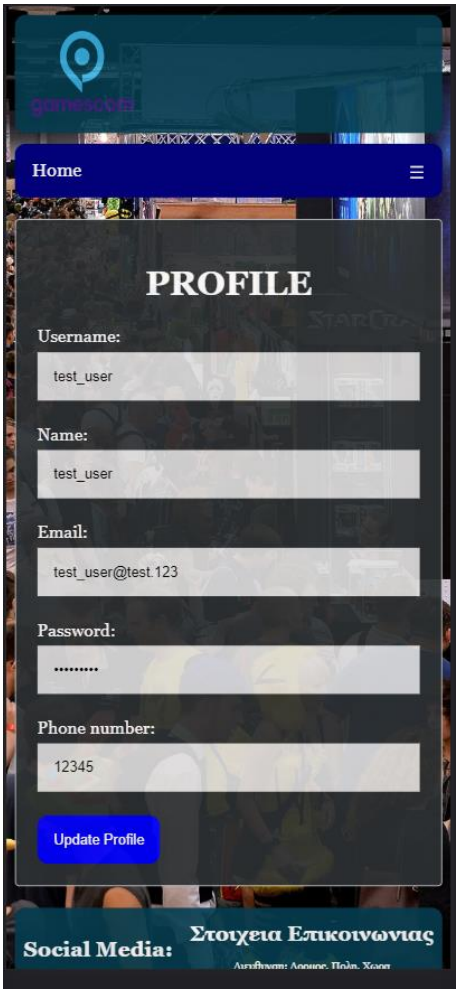
(4 User Profile:

If the user has successfully logged in, their username will be displayed in the top navigation bar. By clicking on it, they can access their profile.



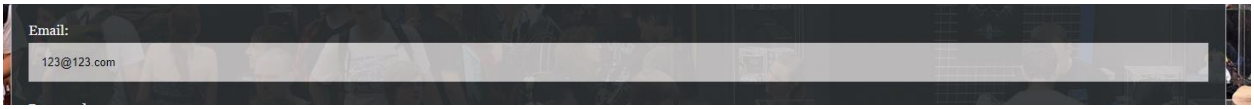
Desktop





Tablet *Smartphone*

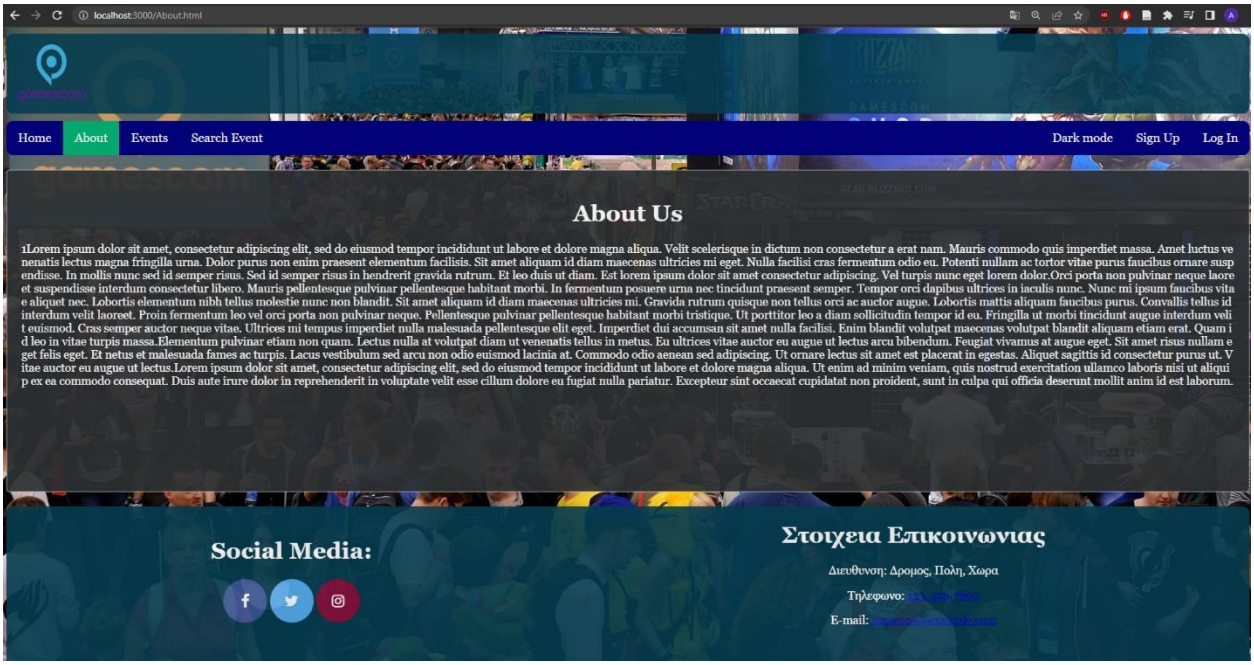
In the profile, all the user's information will be displayed within a form, which can be fully edited except for the "username" field. For example, by changing the "email" from "test_user@test.123" to "123@123.com," we can see the change in the database as well.



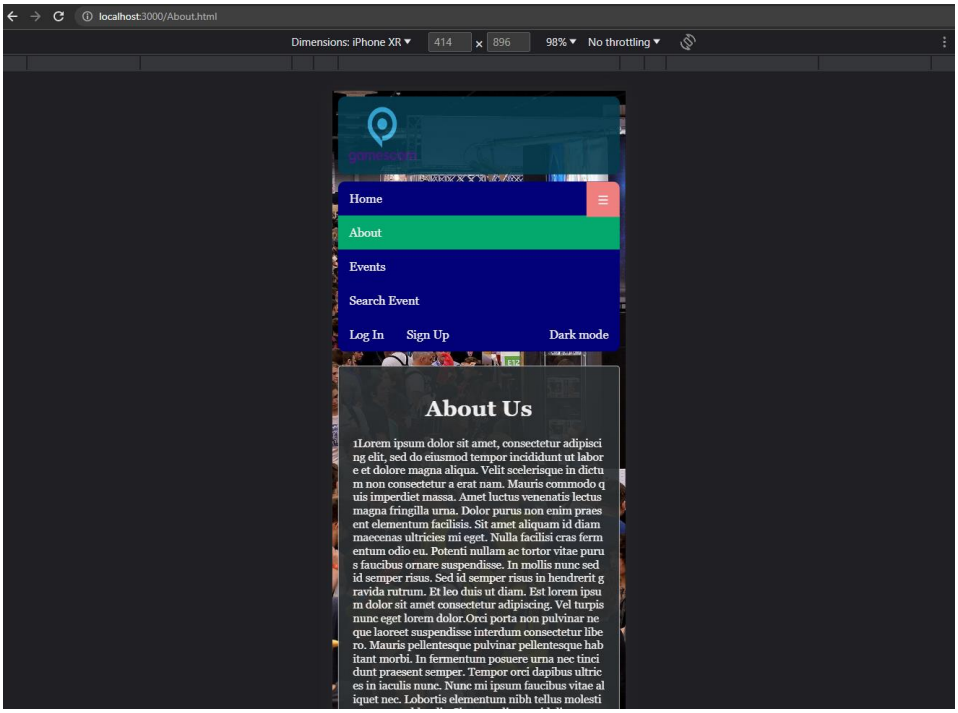

```
_id: ObjectId('649829a53d0d7dd4e5e7e5e0')
username: "test_user"
name: "test_user"
email: "123@123.com"
password: "test_user"
phone: "12345"
```

(5) Information Page:

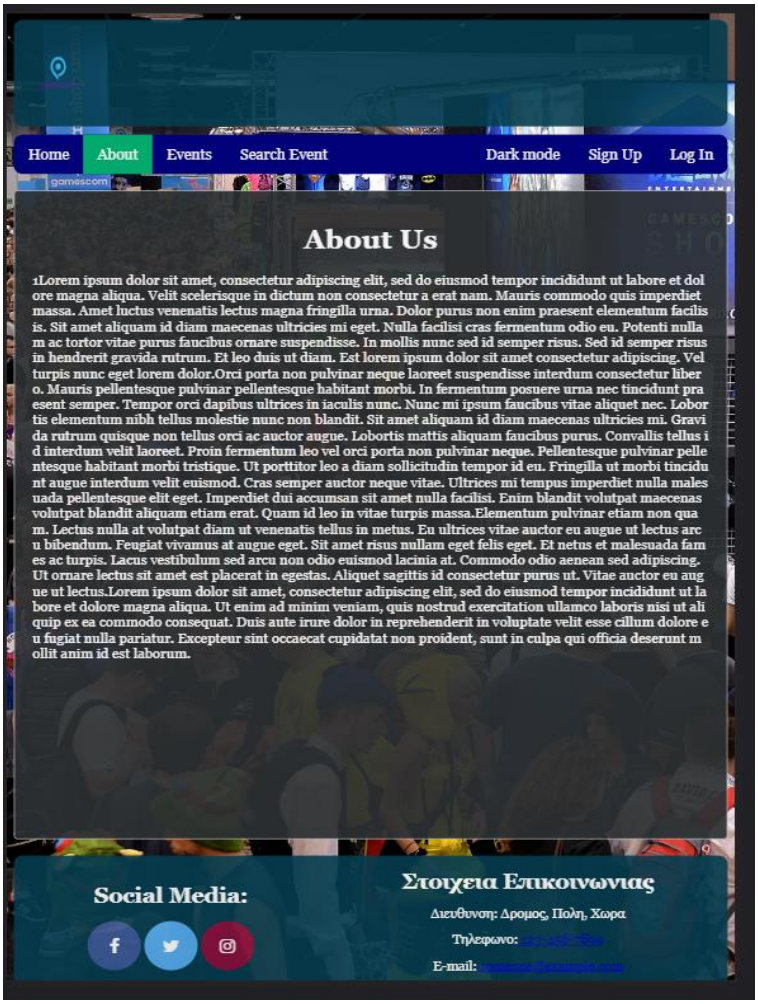
Let's assume that the user will navigate through the pages in the order they appear. Initially, they will click on "About" in the navbar, which will redirect them to the page with general campaign information.



Desktop



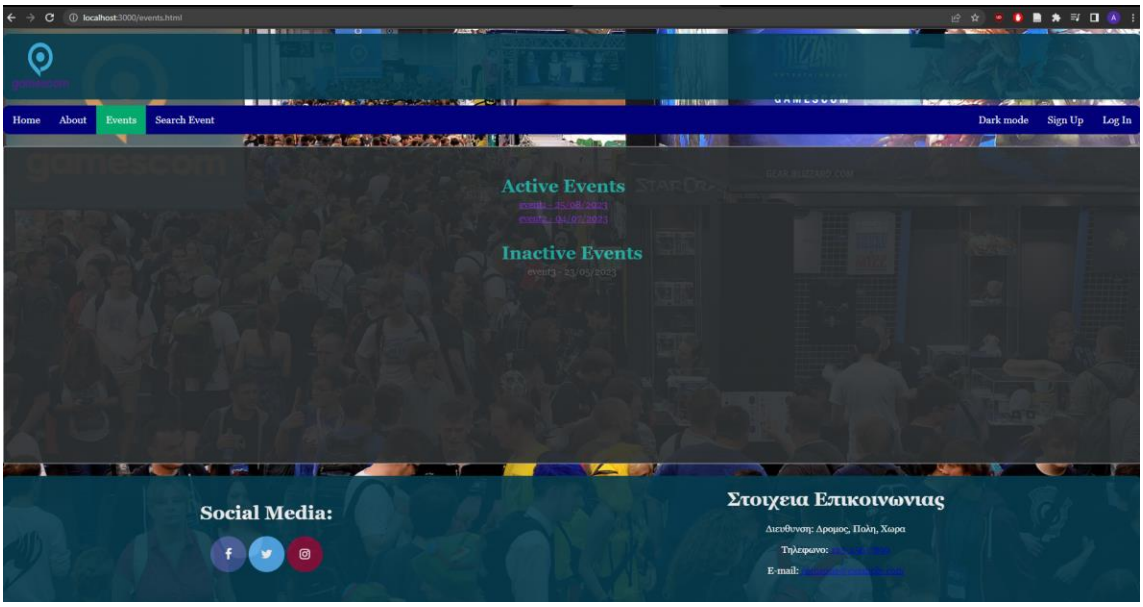
Smartphone



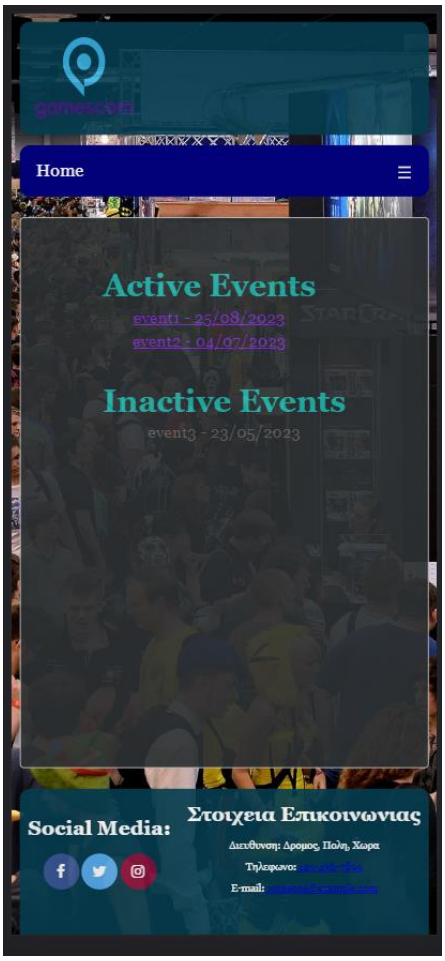
Tablet

(6) Events Page:

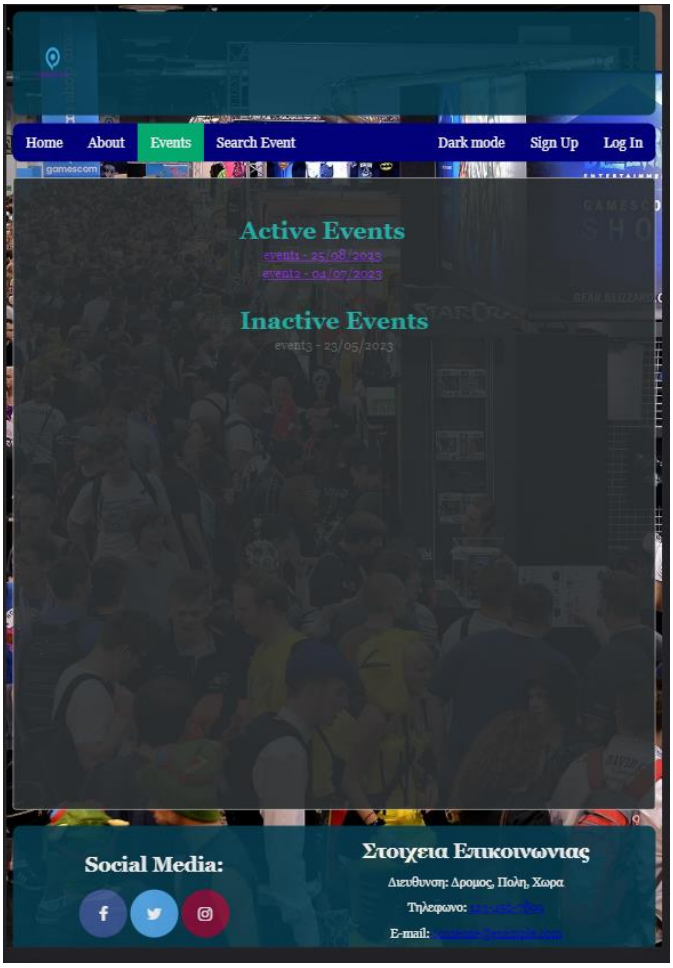
The next page is the "Events" page. On that page, the user can see all the events organized by the campaign, both upcoming and past events.



Desktop

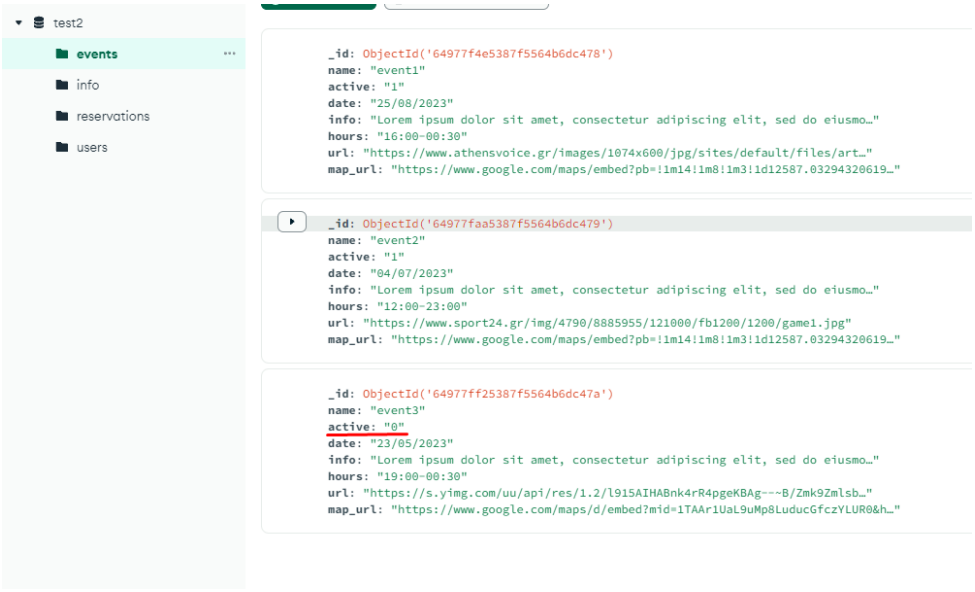


Smartphone

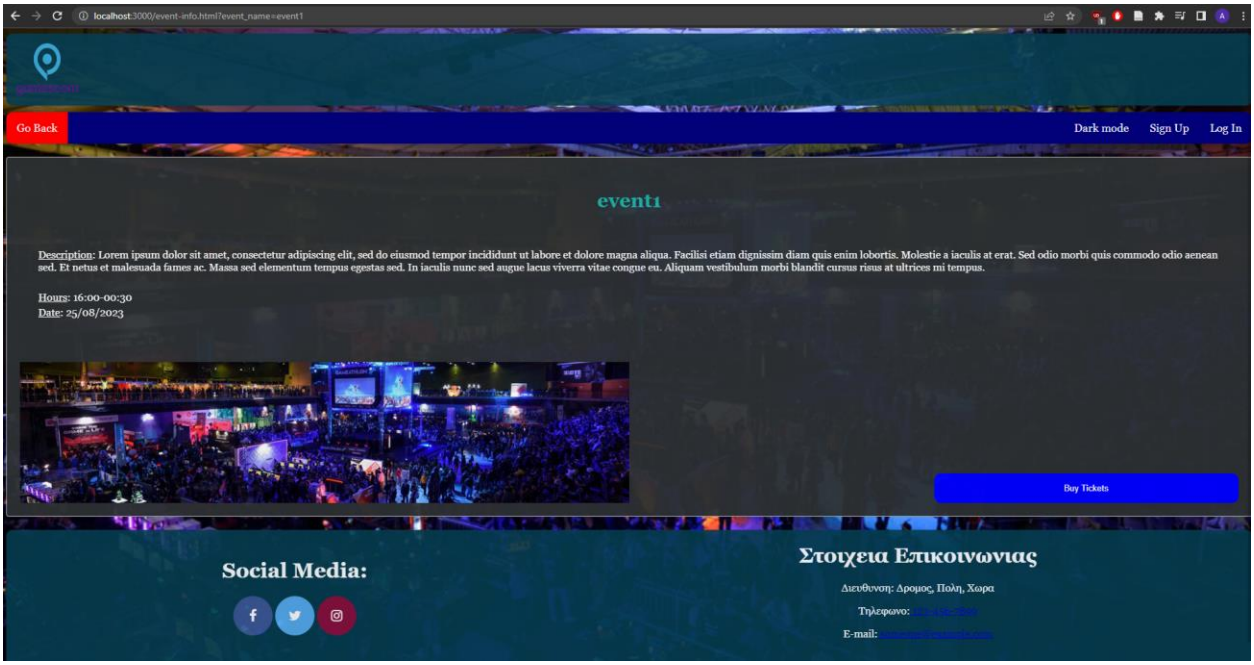


Tablet

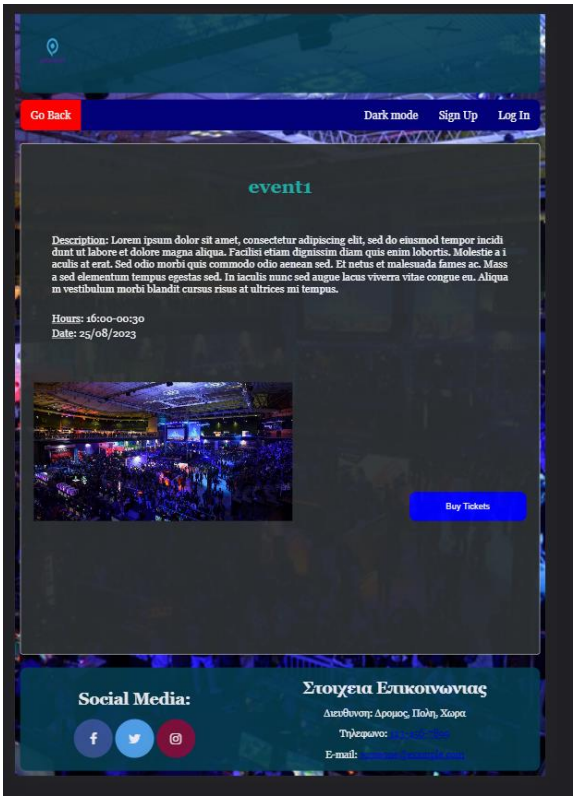
As examples, we have three documents in our database related to events, with "event3" no longer being active. Below are the events in the database collection.



On this particular page, the user can click on an event, which will redirect them to the page with information about that specific event. For example, by clicking on "event1," they will go to the page "/event-info.html?event_name=event1," passing the event name as a parameter to retrieve all the information from the database.



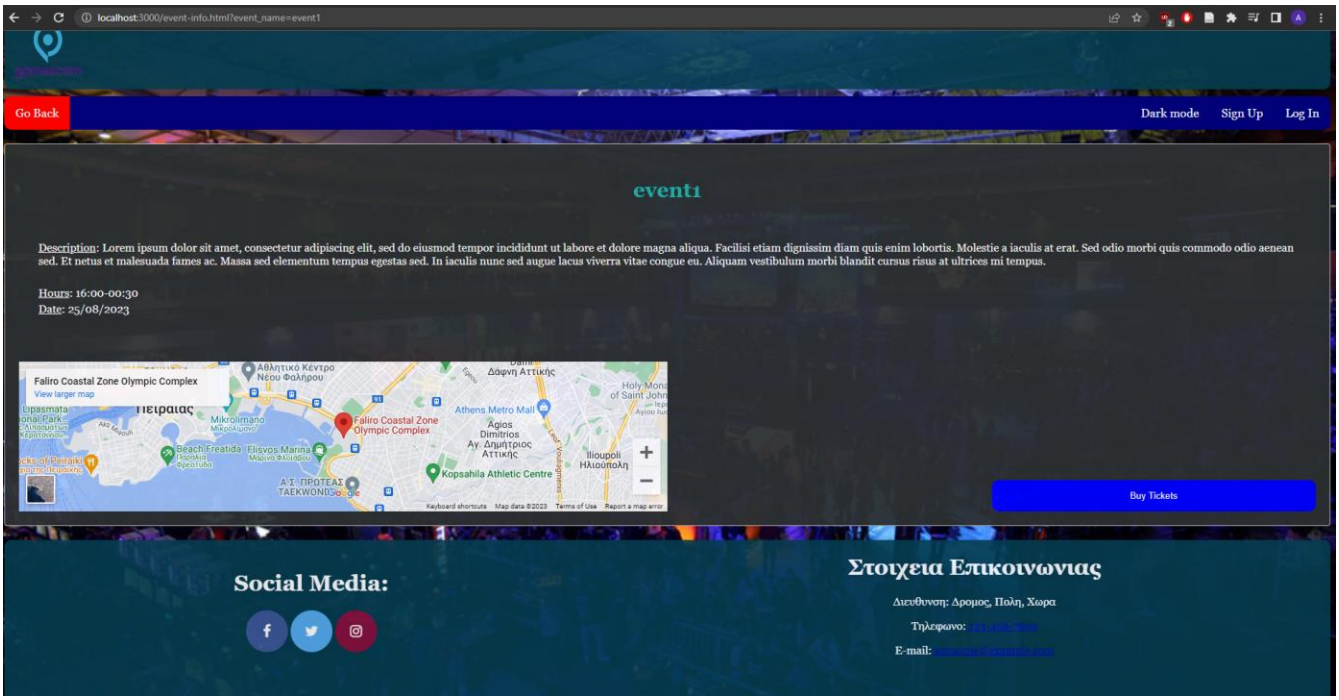
Desktop



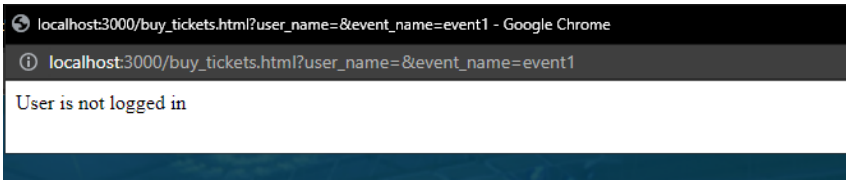
Tablet

Smartphone

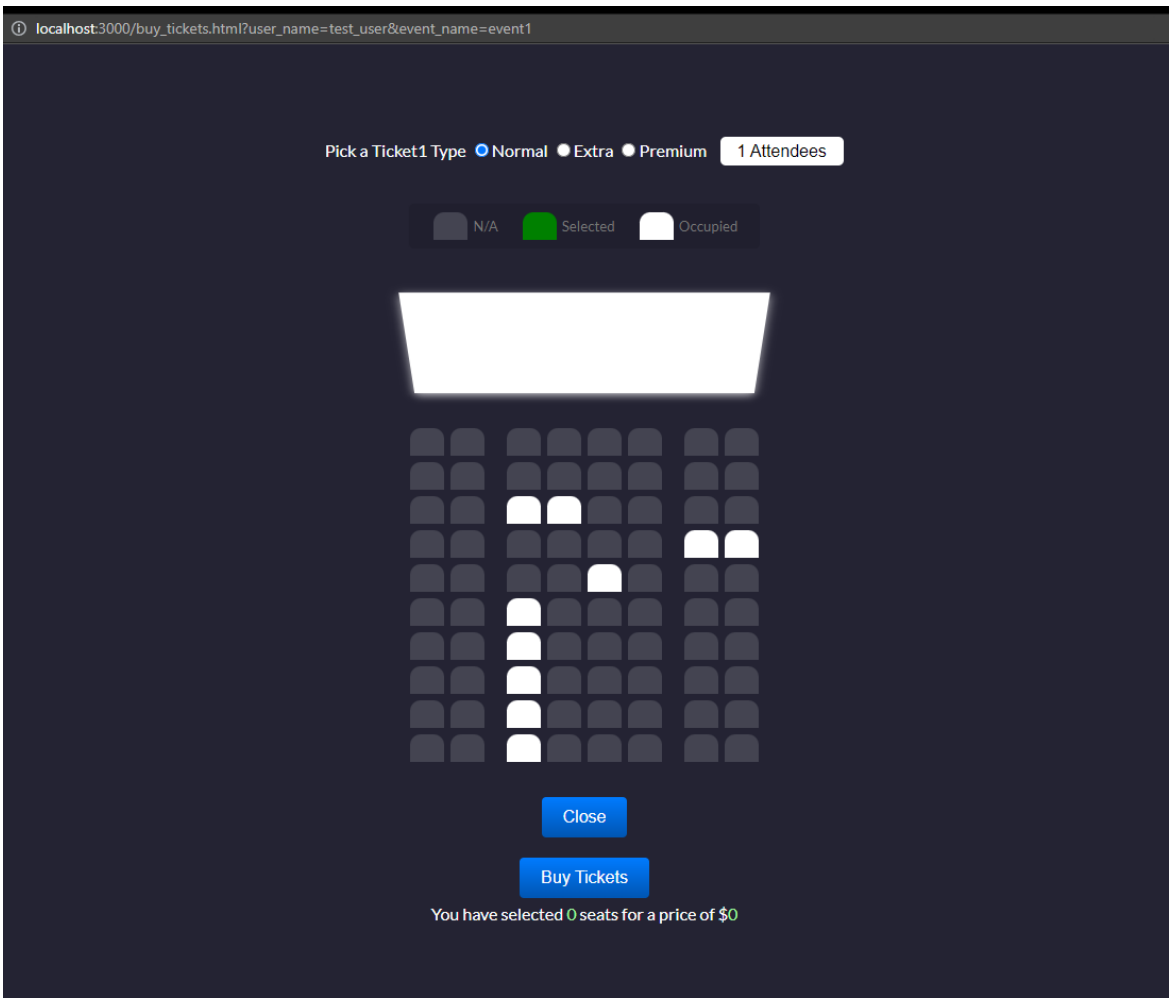
Also, by hovering over an event image, the user can see the location map of the event, with a red pin indicating its exact location



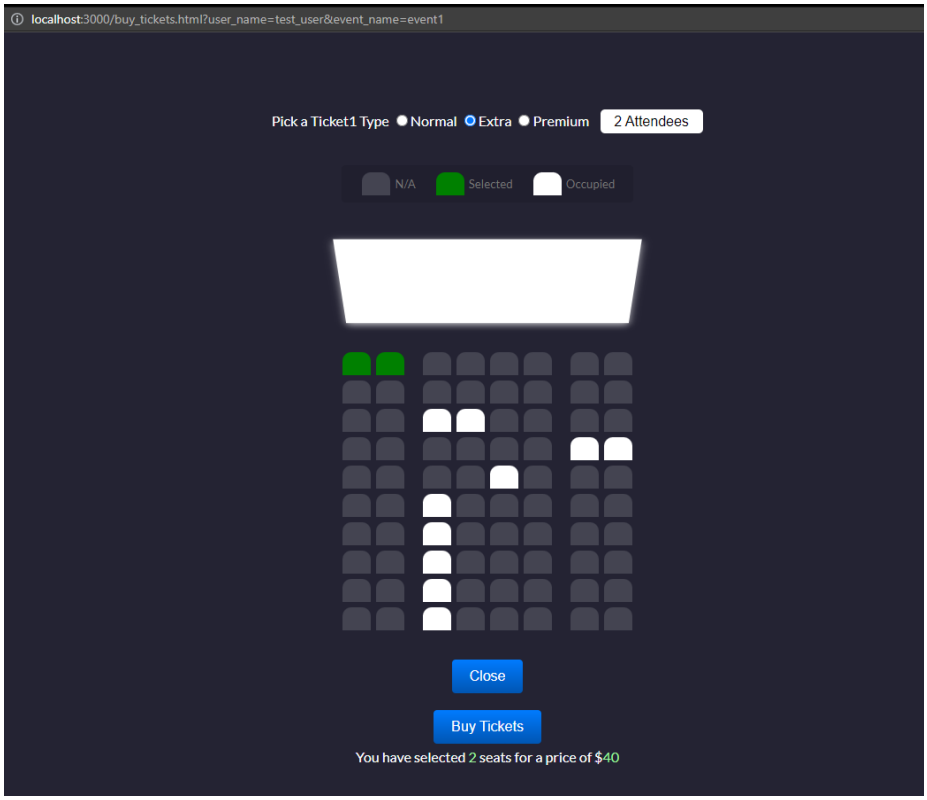
Finally, there is a "Buy Tickets" button that, when clicked, will redirect the user to the page where they can purchase tickets for the specific event if it is active. However, the user needs to be logged in first, so if they are not logged in, they will see the following message:



If the user is logged in and clicks the button to buy tickets, a new window will open where they can select the ticket type. This selection will affect the price and the number of attendees. Additionally, by clicking on the available seats, they can choose the desired seats.



For example, the user "test_user" selects the ticket type "extra" that costs $20, chooses 2 attendees, and selects the first 2 seats.

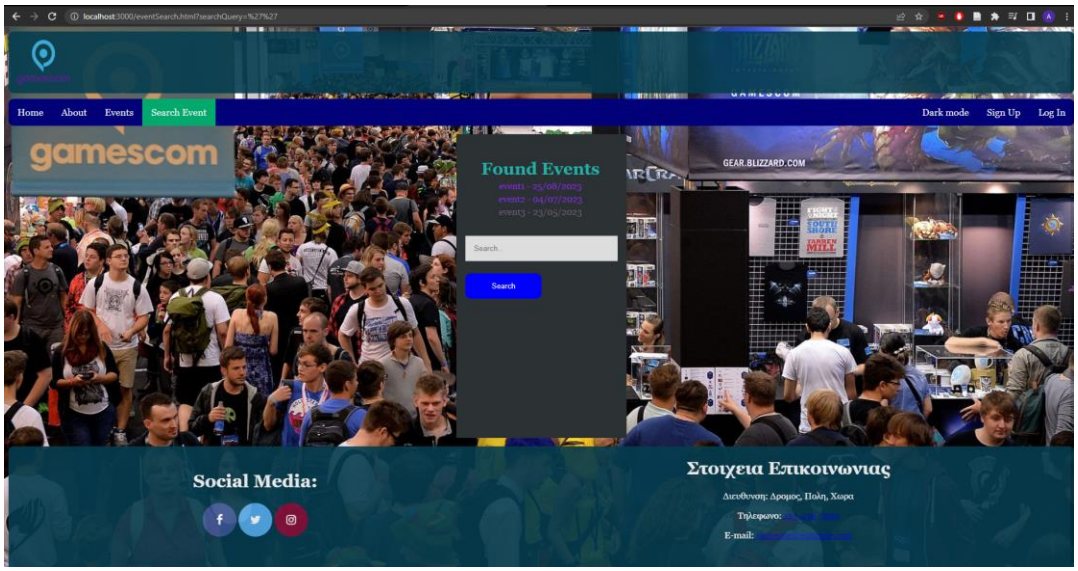


This will create a reservation with the following details:

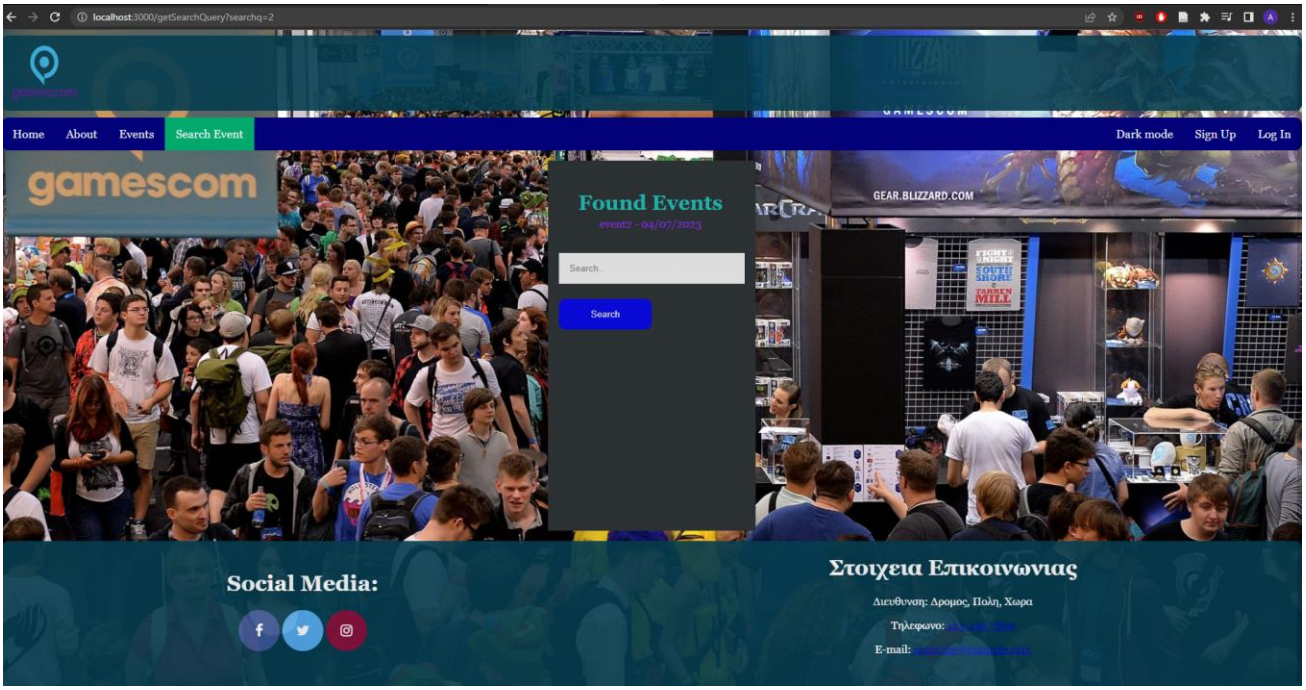

```
_id: ObjectId('6498322d3d0d7dd4e5e7e5e1')
id: "8dedac78-7629-4a47-8cad-2966575418dc"
▾ seats: Array
    0: 0
    1: 1
username: "test_user"
eventname: "event1"
total_price: 40
ticket_amount: 2
ticket_type: "extra"
```

(8) Event Search:

Next, the user can go to the "Search Events" page. There, they can see all the events



On this page, the user can view the active events, which are displayed with a lighter color. By clicking on an event, they will be redirected to the page with information about that event. Additionally, there is a search bar where they can enter a part of the event name they want to find, and the corresponding event will be displayed. For example, entering the number "2" will display "event2":



This happens as the input from the user is taken as a parameter and searched in the events collection for matching names. The matching names are stored in a variable, and the page is rendered again.

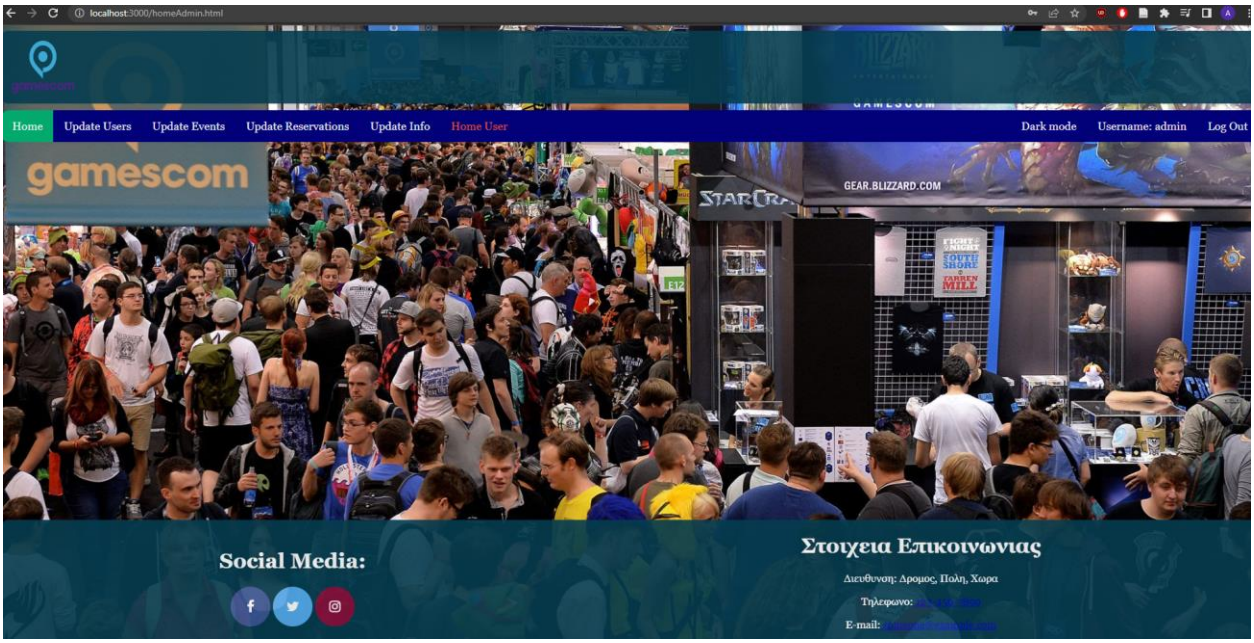
Admin Section:

(10) + (11) Admin Dashboard & Admin Login:

To log in as an admin, they need to enter their credentials. Specifically:

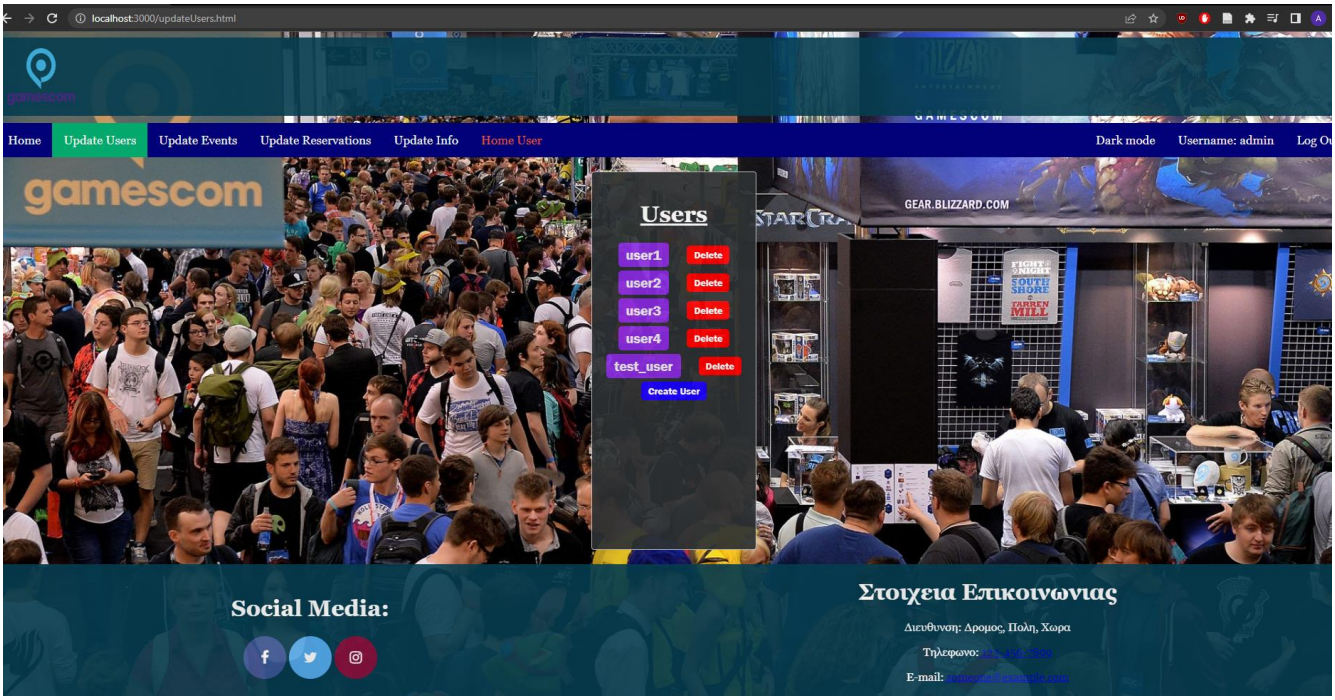
```
_id: ObjectId('6493485b207bab2c8c08a2ec')
username: "admin"
name: "admin"
email: "admin@admin.com"
password: "admin123"
phone: "12345"
```

Once logged in, they will be directed to the admin dashboard.



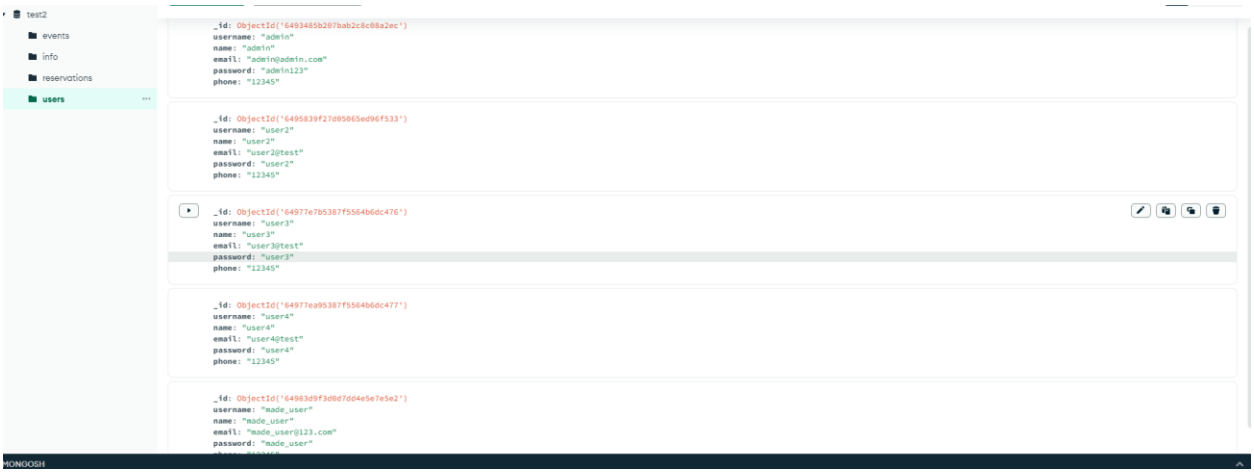
(12) User Editing Page:

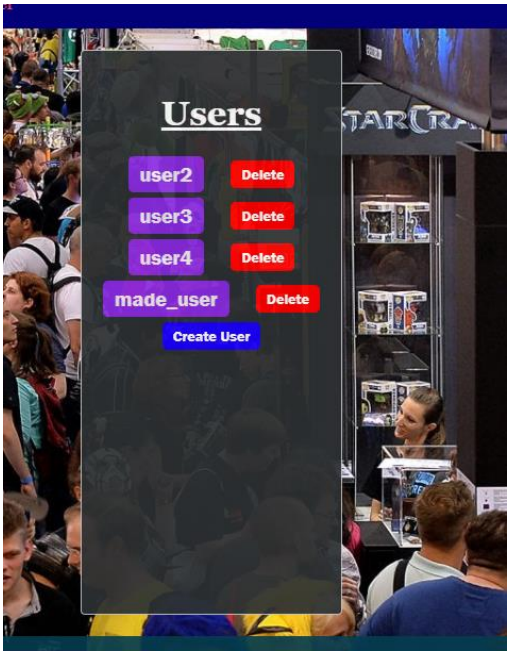
When the admin clicks on "Update Users," they will be redirected to the page where all users are displayed



By clicking the "delete" button, the corresponding user is deleted. If a user who has made reservations is deleted, those reservations are also deleted. For example, if we try to delete the user "test_user":

There is no longer a "test_user" in the "users" collection.



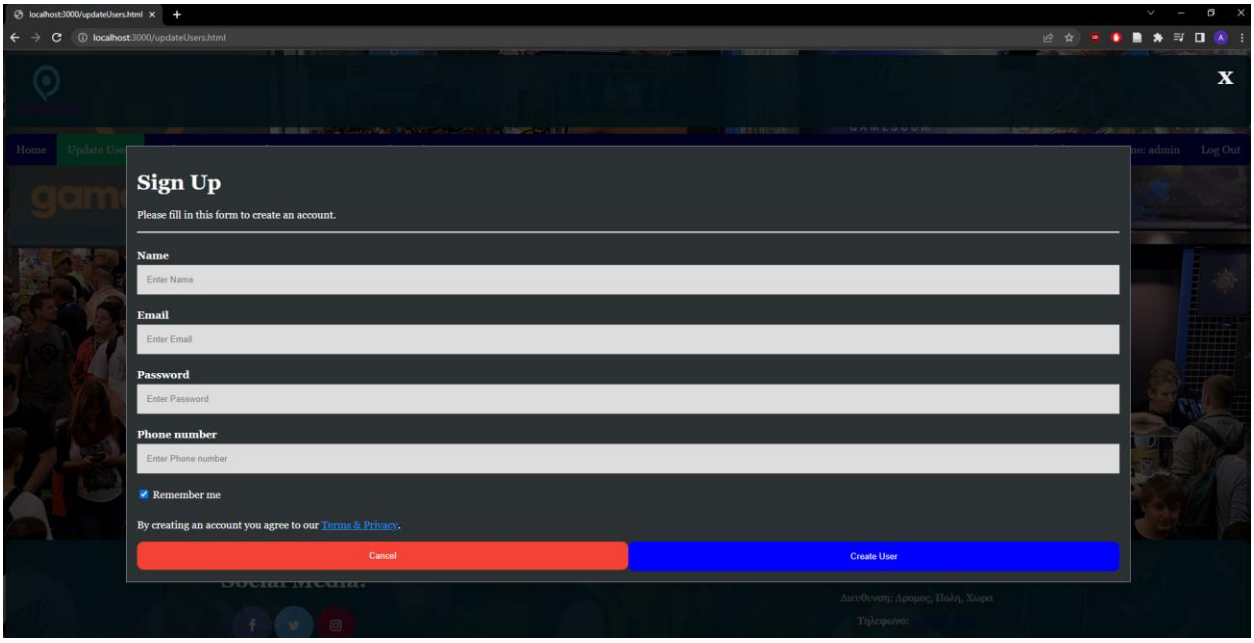


But neither the reservation that the user had created in the previous example exists:

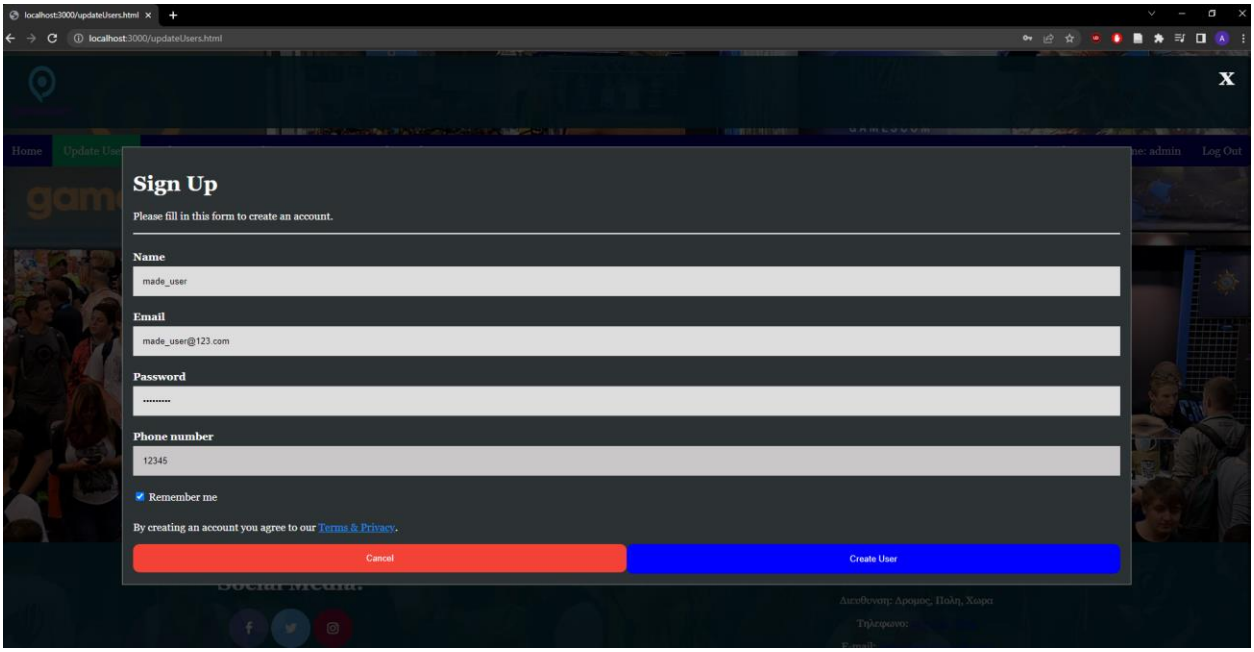

```
_id: ObjectId('6497803d5387f5564b6dc47c')
id: "e467e0b0-8d00-4c69-991f-96a64b42713d"
seats: Array
username: "user2"
eventname: "event1"
total_price: 120
ticket_amount: 4
ticket_type: "premium"
```

```
_id: ObjectId('6497804e5387f5564b6dc47d')
id: "e88a2e40-cbb0-4891-be8d-6b447c461faf"
seats: Array
username: "user3"
eventname: "event2"
total_price: 30
ticket_amount: 1
ticket_type: "premium"
```

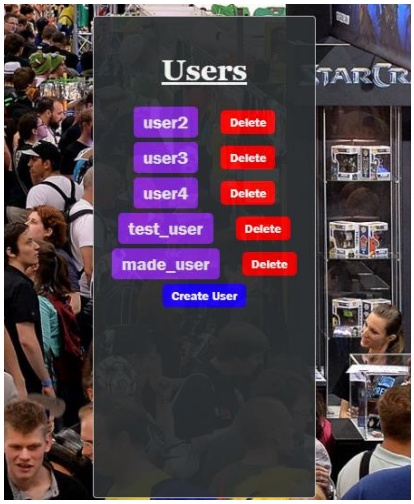
By clicking the "Create User" button, a modal form appears where the admin is prompted to enter the details of the new user they want to create:



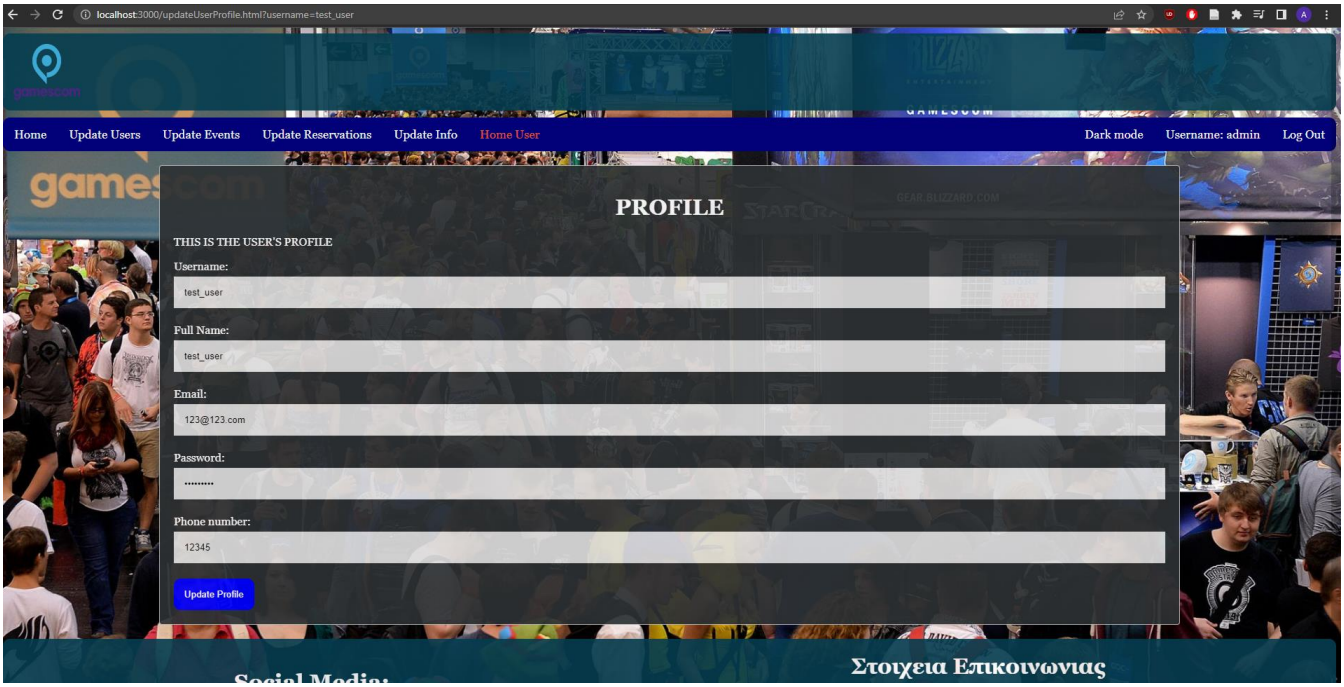
For example, if the admin tries to create a user with the name "made_user":



The user is created, and then the admin is redirected to the "Update Users" page, where the users are refreshed.



By clicking on a user, the admin is redirected to the user editing page.

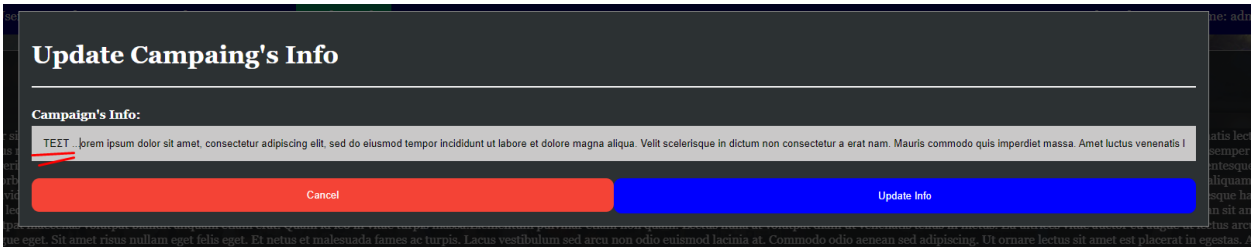


There, a form is displayed with the user's information as stored in the database. By clicking on an input field, the admin can change the information, and by clicking the "Update User" button, the changes are saved in the database. For example, for the user "test_user," changing the "phone number" from "12345" to "123" would result in:


```
_id: ObjectId('649829a53d0d7dd4e5e7e5e0')
username: "test_user"
name: "test_user"
email: "123@123.com"
password: "test_user"
phone: "123"
```

(13 Information Editing Page:

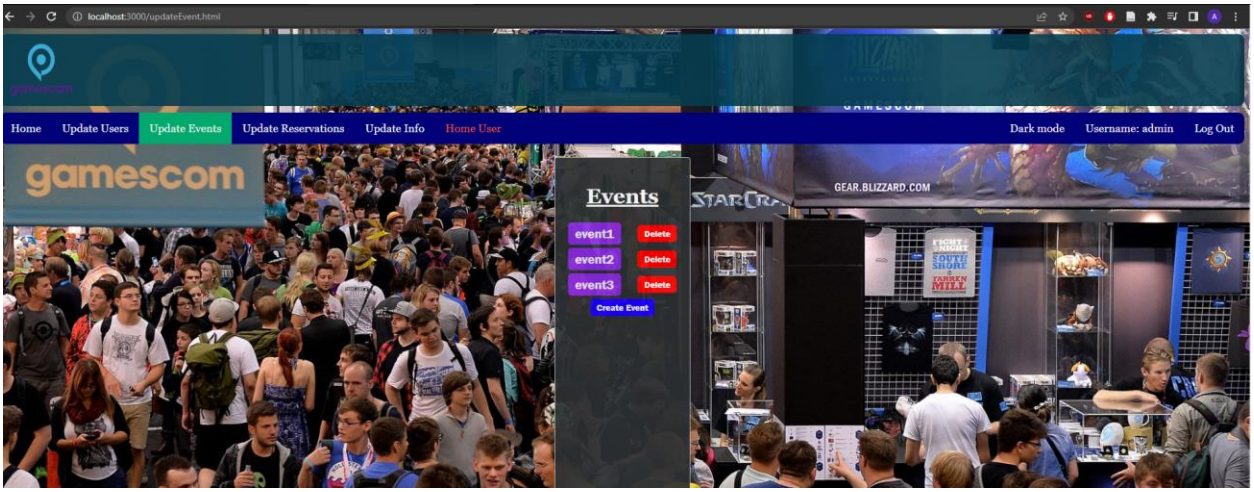
By going to this page, the admin can view the general information about the campaign, which is also displayed on the "About" page. By clicking the "Update Info" button, the admin can edit the information.



And it changes on the "About" page as well

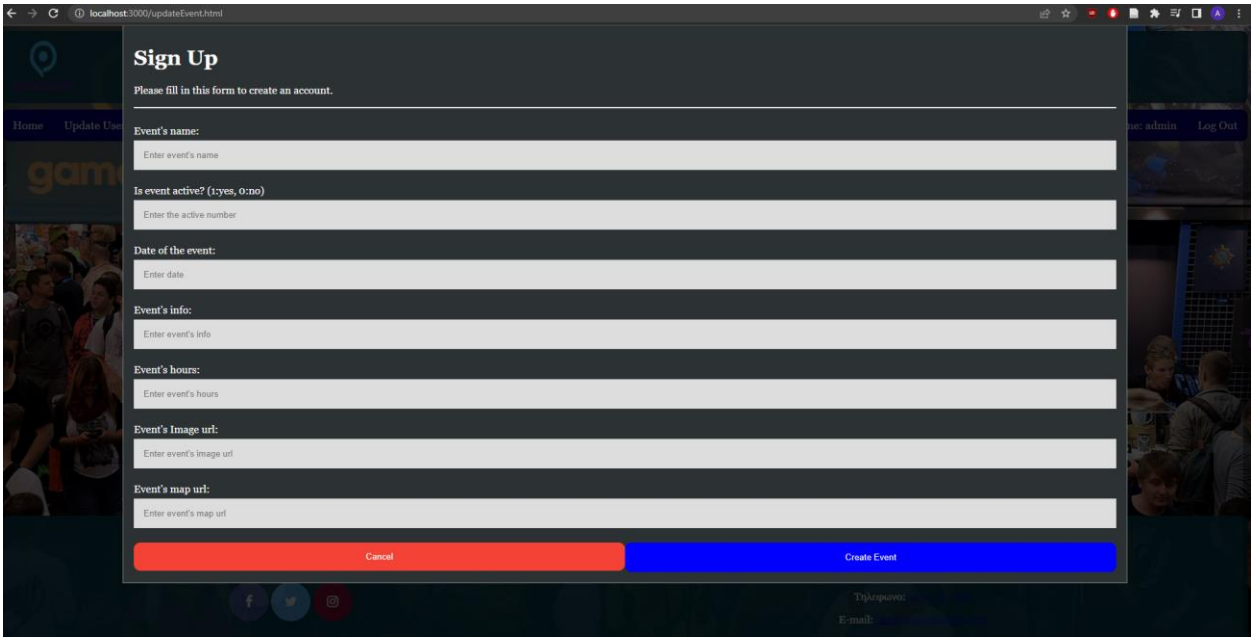
(14) Event Editing Page:

Here, similar to the "Update Users" page, all events are displayed

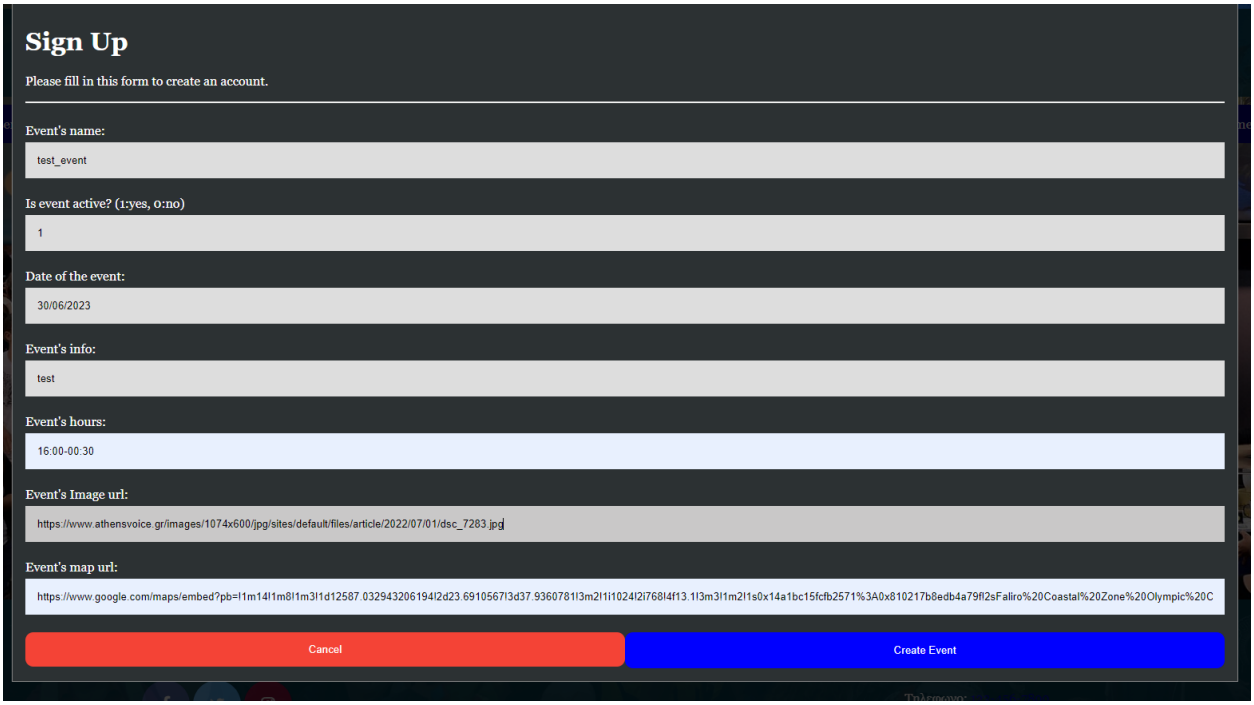


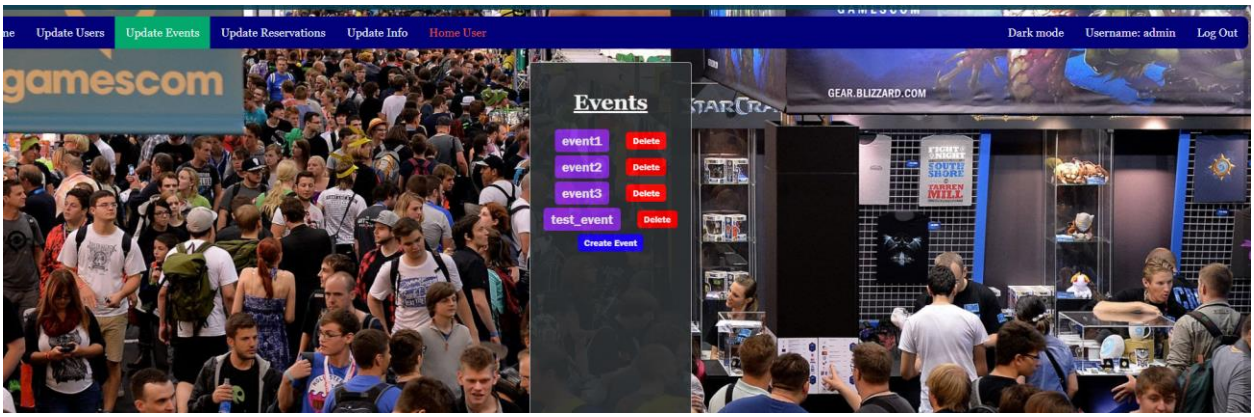
By clicking the "Delete" button, an event is deleted. If an event that has reservations is deleted, the reservations are also deleted.

Finally, by clicking the "Create Event" button, a modal form appears with the necessary fields to create a new event.



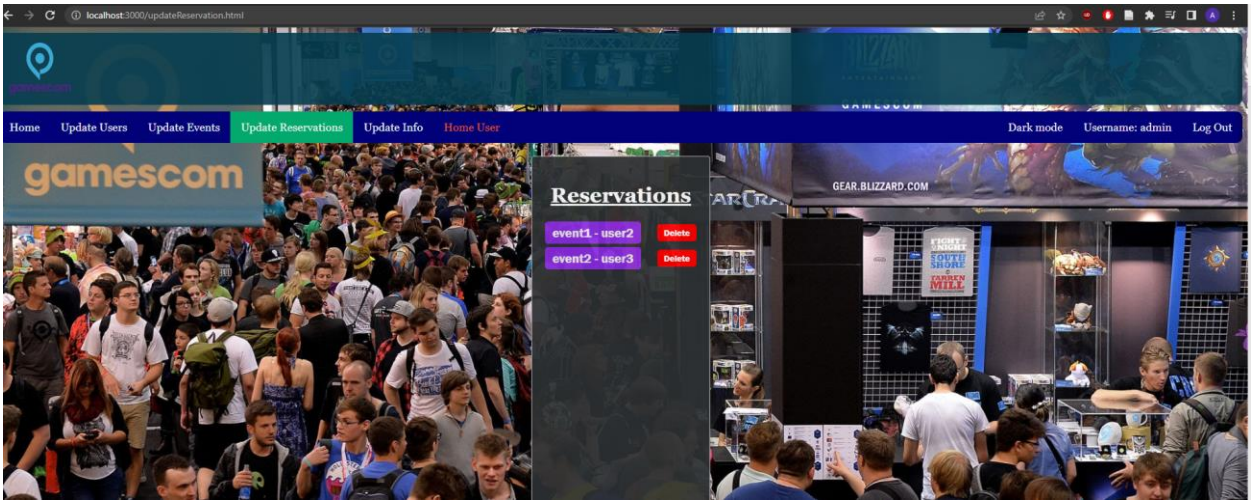
For example (using existing URLs in the database):





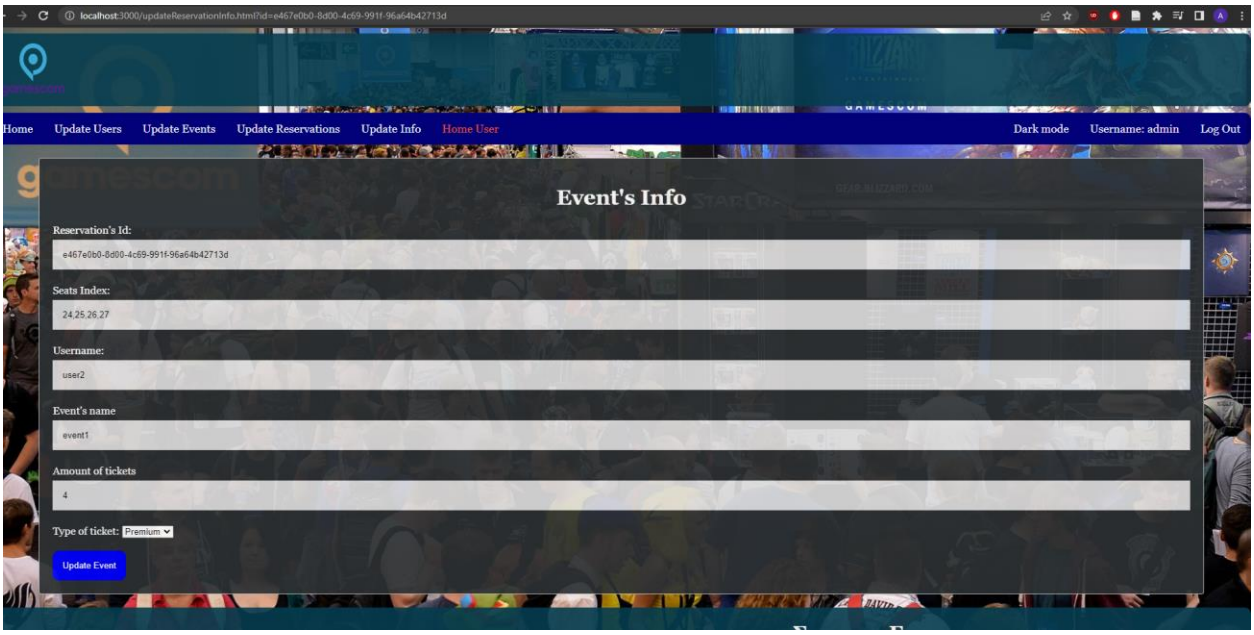

```
_id: ObjectId('649841433d0d7dd4e5e7e5e3')
name: "test_event"
active: "1"
date: "30/06/2023"
info: "test"
hours: "16:00-00:30"
url: "https://www.athensvoice.gr/images/1074x600/jpg/sites/default/files/art…"
map_url: "https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d12587.03294320619…"
```


(14) Reservation Editing Page:



On this page, the admin can see all the reservations that exist, who made them, and for which event.

(Note: The admin cannot create reservations, only delete them and change the ticket type.)



By clicking on a reservation, the admin can view the reservation details. They can change the ticket type, which will also change the final price for the user.

For example, for the 1st reservation "event1 - user2," changing the ticket type from "premium" to "normal" should change the price from $120 to $40, considering that the reservation is for 4 seats, specifically seats 24, 24, 26, and 27.

So initially we have:

```
_id: ObjectId('6497803d5387f5564b6dc47c')
id: "e467e0b0-8d00-4c69-991f-96a64b42713d"
▸ seats: Array
username: "user2"
eventname: "event1"
total_price: 120
ticket_amount: 4
ticket_type: "premium"
```

Into:

```
▸
_id: ObjectId('6497803d5387f5564b6dc47c')
id: "e467e0b0-8d00-4c69-991f-96a64b42713d"
▸ seats: Array
username: "user2"
eventname: "event1"
total_price: 40
ticket_amount: 4
ticket_type: "normal"  40
```

(15 Switching between Admin and User Pages:

By clicking on "Home User," the admin can switch to the user pages that will be presented on the topnav bar, allowing them to access those pages and see any changes they make.

Additionally, when the user pages are displayed, the "Home Admin" option appears, which will show the admin pages again.