Tango Event Management Website

Adam Albert

Student number: n11406119

Course Number: IAB207

Abstract

Tango Event Management is a web application to facilitate and manage events focused on gaming and gaming communities by allowing gamers of different areas to attend different types of events such as but not limited to Competitive, Casual and Social Style events to cater towards the user's specific niche.

Usually, game focused events are organized through small communities through social media or other such means by allowing this tool to be used by the general public to create and manage events whether they are a small get together or a paid entry convention. The aim of this web application will be to centralize events for such communities.

Section 1: technology context

There are many ways to deploy an application to a user, some of these being native applications for mobile devices using IOS or Android, a desktop application or a web application. Before development was started for Tango the advantages and disadvantages of these technologies needed to be compared.

Native applications on mobile devices boast many advantages, they are extremely fast and don't need an internet connection to function as the data is usually stored locally on the device, they are very convenient and user friendly however from a development perspective of this application a native mobile application could be very expensive both in time and capital investment by designing for cross platforms and paying for the application to be pushed to some app stores.

The same can also be said for desktop applications while they have advantages in performance in the context of this situation the disadvantages take over. For desktop applications the application itself will only be used on the machine it's installed on and won't be as convenient as other available technologies, updating and designing for different versions and operating systems will be cumbersome and ineffective.

As Tango is an event management application there are a few key things needed for successful deployment. The application needs to be lightweight, prototype and developed at a very fast pace and it should also not be cumbersome for any user on any different platforms. For these reasons it's clear to see that a web application will be the most advantageous for the needs of tango. By deploying tango as a web application all the listed above needs will be met.

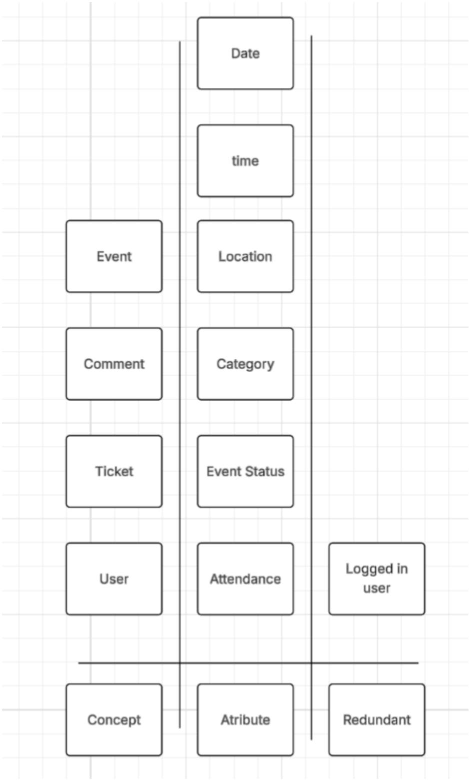
Section 2: User Stories

User stories are used to enable the transition from the real user needs to the technical tasks of the website by breaking apart the broader functions into smaller, more manageable goals of the user. This will be easier to see any expectation that could have been overlooked, below are the user stories for the tango website and the acceptance criteria that will later be refined into the website's object and relations between.

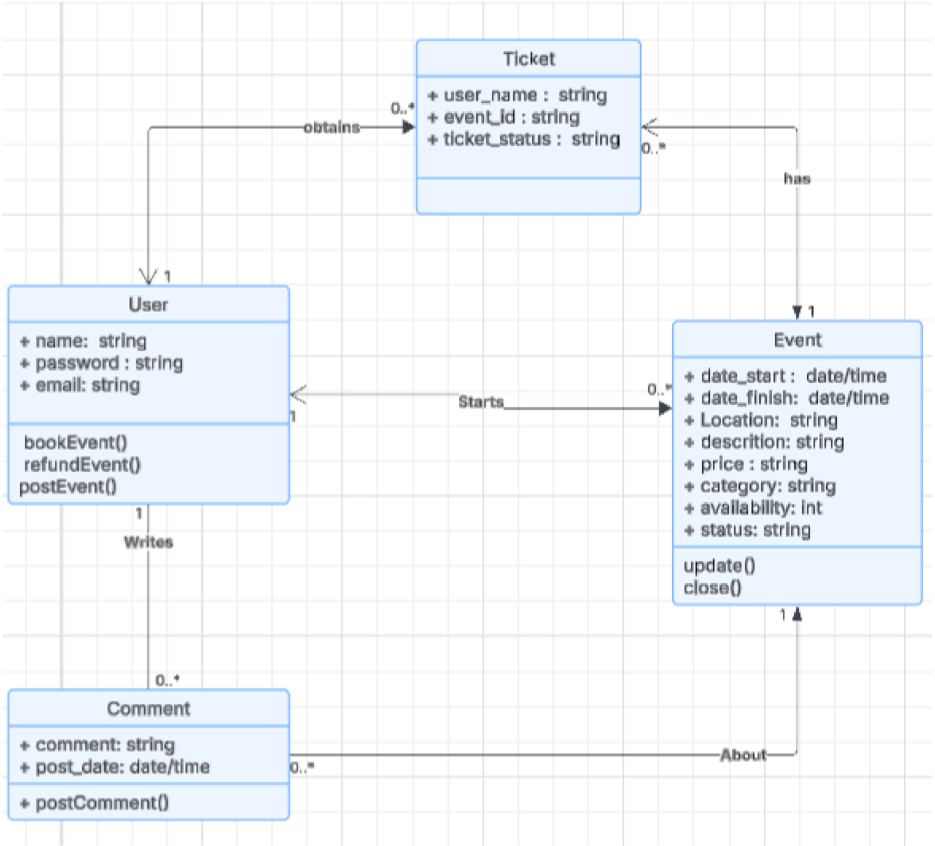
Storie	Description	Acceptance criteria
Browse events	I want to browse different types of new events in my area, including the date and location of the events	• Provide display of **events** with **Location**, **Date** and **availability** • Tagged **events** based on **category** for filtering of events (Competitive, Casual, Social)
See event details	I want to see the details of an event that has caught my eye. This page should have things like date and time, location, cost of an event	• A single view containing all relevant information for the **event** such as **description**, **category**, **date/time, venue**, and **availability**
View tickets	I want to view my current and previous tickets, I will need them later to sign into or for documentation	• Ticket page containing **upcoming and previous events** that the user has attended
Book event	I want to book an event that will be added to my tickets	• Ability to add **ticket** to account
search event	I want to search for a specific event that my friend told me about	• Search bar that will find and **event** the user is looking for
Comment on event	I want to comment on an event that I went to because I had a great time and want the organizers and other people to know	• **Comments** section in event details page that the **user** can post into
Post event	I want to post a social game event to the website for people in my area	• Event management page that will enable **logged in users** to post events by categories
Update event	I have made a mistake in how many people wanted to come and need to update the capacity and venue of the event	• Ability to **update event** details after posting
Cancel event	I want to cancel or refund the ticket that I had booked because I cannot make it anymore	• Refund function for transactions or general cancelation of **attendance**

Section 3: Conceptual Modal

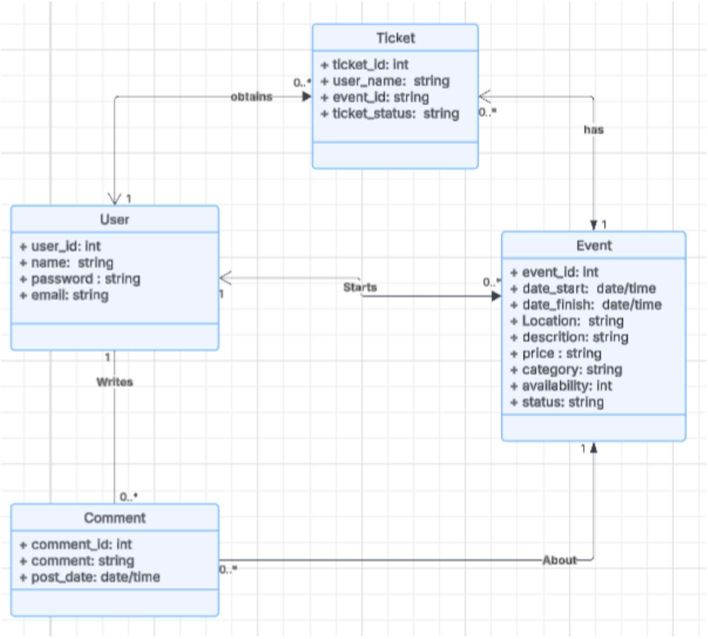
By separating the user stories acceptance criteria into the core concepts of the website it becomes easier to visualize how different components of the website interact with one another. This helps visualize the structure of the website showing not just individual components but their relationships and dependencies allowing for a more consistent design.



Further following this process, the flow of the concepts and associating attributes can be visualized to produce the resulting conceptual modal



Section 4: Data Model



Appendix 1: Use of AI Declaration in IAB207

I confirm that I used AI in an authorized way (as described by the assignment brief).

Summary of AI use

(no word limit but we suggest 50-100 words describing how you used AI)

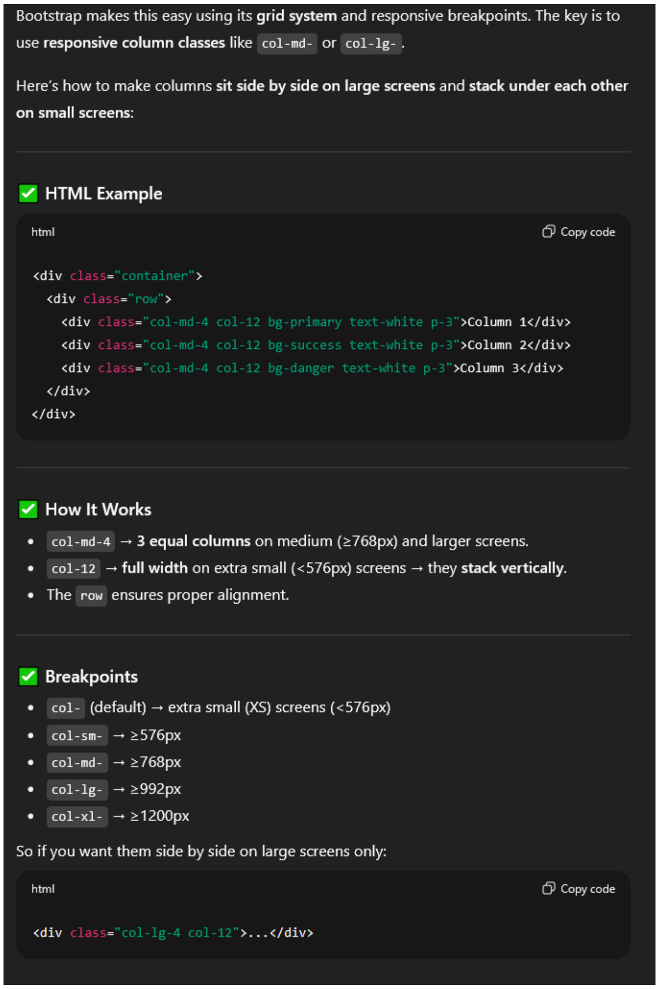
I used ChatGPT to clarify coding mistakes or formatting misunderstandings with bootstrap components as well as some image generation

Key examples of AI use

Your marker needs to clearly understand how you used AI to improve your work. Provide representative samples of how you used AI using the requested format.

{

How can I make bootstrap columns collapse under each other at a small breakpoint



Bootstrap makes this easy using its **grid system** and responsive breakpoints. The key is to use **responsive column classes** like `col-md-` or `col-lg-`.

Here's how to make columns **sit side by side on large screens** and **stack under each other on small screens:**

✅ HTML Example

```html
<div class="container">
  <div class="row">
    <div class="col-md-4 col-12 bg-primary text-white p-3">Column 1</div>
    <div class="col-md-4 col-12 bg-success text-white p-3">Column 2</div>
    <div class="col-md-4 col-12 bg-danger text-white p-3">Column 3</div>
  </div>
</div>
```

✅ How It Works

- `col-md-4` → **3 equal columns** on medium (≥768px) and larger screens.
- `col-12` → **full width** on extra small (<576px) screens → they **stack vertically**.
- The `row` ensures proper alignment.

✅ Breakpoints

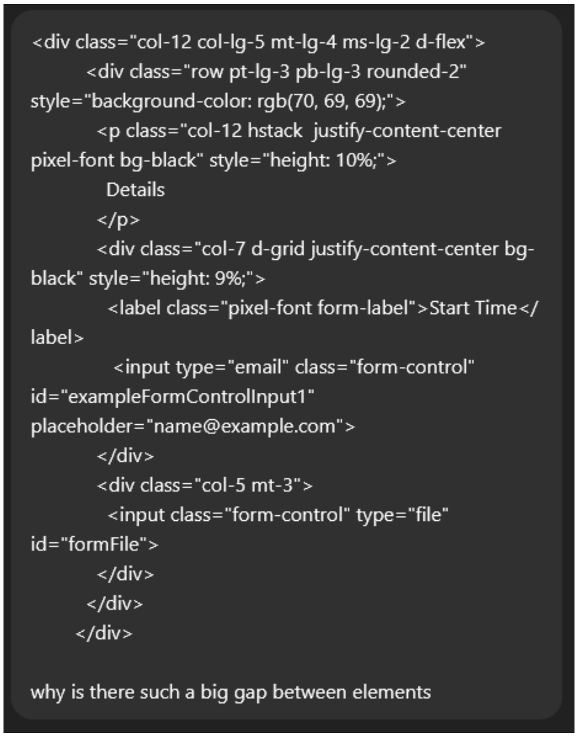
- `col-` (default) → extra small (XS) screens (<576px)
- `col-sm-` → ≥576px
- `col-md-` → ≥768px
- `col-lg-` → ≥992px
- `col-xl-` → ≥1200px

So if you want them side by side on large screens only:

```html
<div class="col-lg-4 col-12">...</div>
```

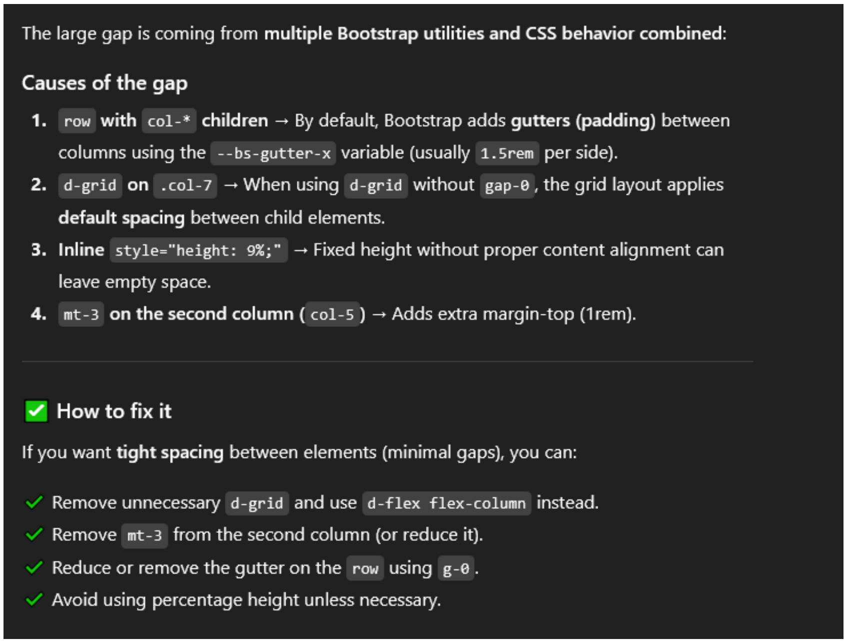
======================== NEXT SAMPLE OF AI USE ====================

{



```
<div class="col-12 col-lg-5 mt-lg-4 ms-lg-2 d-flex">
        <div class="row pt-lg-3 pb-lg-3 rounded-2"
style="background-color: rgb(70, 69, 69);">
          <p class="col-12 hstack  justify-content-center
pixel-font bg-black" style="height: 10%;">
            Details
          </p>
          <div class="col-7 d-grid justify-content-center bg-
black" style="height: 9%;">
            <label class="pixel-font form-label">Start Time</
label>
             <input type="email" class="form-control"
id="exampleFormControlInput1"
placeholder="name@example.com">
          </div>
          <div class="col-5 mt-3">
            <input class="form-control" type="file"
id="formFile">
          </div>
        </div>
      </div>
```

why is there such a big gap between elements



The large gap is coming from **multiple Bootstrap utilities and CSS behavior combined:**

Causes of the gap

1. **`row` with `col-*` children** → By default, Bootstrap adds **gutters (padding)** between columns using the `--bs-gutter-x` variable (usually `1.5rem` per side).
2. **`d-grid` on `.col-7`** → When using `d-grid` without `gap-0`, the grid layout applies **default spacing** between child elements.
3. **Inline `style="height: 9%;"`** → Fixed height without proper content alignment can leave empty space.
4. **`mt-3` on the second column (`col-5`)** → Adds extra margin-top (1rem).

✅ How to fix it

If you want **tight spacing** between elements (minimal gaps), you can:

✓ Remove unnecessary `d-grid` and use `d-flex flex-column` instead.
✓ Remove `mt-3` from the second column (or reduce it).
✓ Reduce or remove the gutter on the `row` using `g-0`.
✓ Avoid using percentage height unless necessary.