Hello everyone. This is our Project: Zenyu

Group Members: Denysse Cunza, Shariar Kabir, Angela Lim, Gillian Pantaleon

INTRO:

Today we will be demoing our app Zenyu.

Before we begin, we would like to remind you guys about our product definition. Our product definition is as follows: In order to encourage individuals to lead a healthy lifestyle, our product will solve the average person's problem of struggling with overwhelming emotions such as stress or anxiety by giving them a user-friendly web application that provides a structured approach to journaling for guidance and offers safe boundaries. We will know if our product works when we see improvement of mood over time as people continually use the app.

DEMO:

Now we will move onto the demo.

HOMEPAGE DEMO:

To begin we have our homepage. This is the homepage visitors and guests see. It includes information about Zenyu, our project mission and some of our app features. Here we are showing the different ways to use the website, we have a journal so that they are able to express themselves. We offer guided prompts so that the user can choose what they want to write about. There is a mood tracker that keeps a daily log of what the user is feeling. The user can also add pictures in order to customize their entries. We included our story, to explain why we created this website. We decided to do a single page application because it is in trend and it is easier for the user to navigate. From this visitor homepage, we have two buttons that will redirect us to the login and sign up pages.

REGISTER + LOGIN DEMO:

When the user clicks on the sign up button, they will be redirected to our register page where they can create a new account. We will be demonstrating how to create an account now. After making a new account, they will be directed to the login homepage where they can use the newly made account. And now Gillian will show how to log in and how to use the journal. When the user logs in, they will also be directed to this homepage because only registered users can see it.

USER HOMEPAGE DEMO:

This homepage is where users can create a new journal and look back at old journal entries. So when the user logs in and they haven't made a journal entry for the current day, these four cards will appear with some prompts from our database. These prompts are here to give the user some ideas to write about. They can then select one of these prompts which will direct them to the journal entry form.

JOURNAL ENTRY FORM DEMO:

On this page, the user will be able to make their entry. We have a rich text editor so the user can change the font and how the text is formatted. They will also be able to choose a mood that best describes their day or their journal post. Once they click submit, they will then be redirected back to the homepage.

BACK TO USER HOMEPAGE DEMO:

Now that the user has made a post, the four cards are no longer rendered and instead, we have the journal entry. We can change the mood if they submitted one as shown here. We can also change the text and delete the journal as well. If the user wants to look back at their previous posts, they can just click a date on the calendar. If they go back to a date that has no post, they will have this message appear.

ALL USER PROMPTS:

The user is also able to look at their past journal prompts by clicking the library button on the navbar. Here we have a table of past prompts the user has done, when they did it and the category it is part of.

LOGOUT DEMO:

Once the user is done, they can click the logout button on the navbar which will redirect them to the guest homepage. And that concludes our demo

EXTRA MATERIALS:

For our tech stack we will be using React for the frontend, C# for the backend and PostgreSQL for the database. In addition, we're using CKEditor for our rich text editor on the frontend and HTML along with CSS for styling.

WORK DISTRIBUTION:

Our team was split up into frontend and backend. Our frontend was Angela, Denysse and Gillian while Shariar worked on the backend. Angela who was also our PM assigned tasks for the team and created the dashboard for when the user logs in and connected our rest APIs to the frontend. Denysse worked on our landing page and the journal creation page. Gillian worked on the design of our app and worked on the login and registration page and connecting those endpoints. Shariar worked on creating the rest APIs for journals, images and prompts, the APIs for login and registration and setting up our database.

CONCLUSION:

And that concludes app presentation. Any questions?

TECHNICAL PRESENTATIONS: UI Design:

Now we will move onto our technical presentations, starting with User Interface (UI) Design.

WHAT IS UI DESIGN?

User Interfaces are the access points where users interact with designs. They can be graphical, voice-controlled, or gesture-based. UI design is the process designers use to create interfaces which users find easy-to-use and pleasurable. UI Design is not the same as User Experience (UX) Design. The terms are used interchangeably, but UI Design is really a subset of UX Design. For example, UX Design is like a car. It covers the entire spectrum of the user experience, which includes branding, usability, functionality, and the design. UI Design is analogous to the driving console *(click for animation)*. UI is more concerned with the surface and overall feel of a design, that is, how the user will interact with the product. Applications have a conventional look and feel because of design patterns. Patterns are reusable components that enable designers to choose the best and commonly used interfaces. There are several paradigms for user interface. We will focus on Jakob Nielsen's Usability Heuristics for UI Design.

THE 10 USABILITY PRINCIPLES:

The 10 Usability principles were developed in the 1990s and are based on years of experience in the field of usability engineering. They've been reflected in many products by Apple, Google, and Adobe and have become rules of thumb for human-computer interaction. These are principles that every designer should know. In the interest of time, we will cover the following (*indicated by blue stars).*

VISIBILITY OF SYSTEM STATUS:

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time. The first goal is to communicate clearly to users what the system's state is. No action with consequences to users should be taken without informing them. The second goal is to present feedback to the user as quickly as possible. Knowing what the current system status is can help users learn the outcome of their

prior interactions and determine next steps. Also, predictable interactions create trust in the product and the brand.

SYSTEM MATCH TO THE REAL WORD:

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order. The key is to ensure users can understand meaning without having to go look up a word's definition. We implemented a calendar because it is easily recognizable and serves the same purpose as its real-life component.

USER CONTROL & FREEDOM:

Users often perform actions by mistake. They need a clearly marked emergency exit to leave the unwanted action without having to go through an extended process. When it's easy for people to back out of a process or undo an action, it fosters a sense of freedom and confidence. Exits allow users to remain in control of the system and avoid getting stuck and feeling frustrated. We implemented a text editor with 'undo' and 'redo' commands. These functions give users freedom because they don't have to worry about their actions.

CONSISTENCY & STANDARDS:

The designer should ensure that terminology is maintained across similar platforms.
There should be **no ambiguity in wording** so that users are certain that words, actions and situations represent the same thing.
Benefits:

- User will not be left in confusion on what action to take
- Will lead to a uniform look across the platform.

RECOGNITION RATHER THAN RECALL:

Designers must minimize the user's memory load by making elements, actions, and options visible.
Users should **not have to remember information across the site**, they must be offered easy means of retrieving this information.
Benefits:

- Avoids making the users remember information.
- Allows for easy readability, and less ambiguity

Humans have limited short-term memories. Interfaces that promote recognition reduce the amount of cognitive effort required from users.

FLEXIBILITY & EFFICIENCY OF USE:

AESTHETIC & MINIMALIST DESIGN:

The designer must reduce clutter in the display by only showing what is necessary for current tasks.
The interface should not contain information which is irrelevant.
Every piece of information in an interface competes with relevant units of information.
Benefits:
Provides clearly visible and unambiguous means of navigating
The user is not overwhelmed with information that is useless.

TECHNICAL PRESENTATIONS: CACHING:

The next topic will be caching, specifically Redis caching. First we will explain what caching is. Caching is the process of storing copies of files in a cache, or temporary storage location, so that they can be accessed more quickly. A cache on the other hand is a temporary storage location for copies of files or data so that in the future, data can be served faster.

WHY USE CACHING:

So why do we use caching? There are a lot of benefits to caching but we will discuss a few of those. One advantage to caching is that it improves application performance. Because memory is orders of magnitude faster than disk (magnetic or SSD), reading data from in-memory cache is extremely fast (sub-millisecond). This significantly faster data access improves the overall performance of the application. Caching also reduces database costs because a single cache instance can provide hundreds of thousands of IOPS (Input/output operations per second), potentially replacing a number of database instances, thus driving the total cost down. This is especially significant if the primary database charges per throughput. In those cases the price savings could be dozens of percentage points. Using caching also reduces the load on the backend. By redirecting significant parts of the read load from the backend database to the in-memory layer, caching can reduce the load on your database, and protect it from slower performance under load, or even from crashing at times of spikes. By

reducing the load on the backend we can also get more predictable performances. A common challenge in modern applications is dealing with times of spikes in application usage. Examples include social apps during the Super Bowl or election day, eCommerce websites during Black Friday, etc. Increased load on the database results in higher latencies to get data, making the overall application performance unpredictable. By utilizing a high throughput in-memory cache this issue can be mitigated.

WHAT IS REDIS?

Now we will discuss a specific data store where we can implement caching called Redis. Redis is an open source in-memory data structure store, used as a database, cache and message broker. It is used by many companies like Snapchat, Twitter, Slack, Uber, Airbnb, and Pinterest.

PROS AND CONS OF REDIS:

For the pros of Redis, it is fast and easy to set up. It has flexible data structures. It also has zero downtime or performance impact while scaling up or down. Finally it is open source and stable. For cons, it is an expensive platform to use and requires a lot of RAM as the cached data is stored in there.

CONCLUSION:

And that concludes the technical presentations. Any questions?