
p
picocanvas
wei_bin_theo
Private
Run
Edit

CS6131 Database Design

Project Design Phase Report Submission

By Ang Wei Bin and Theophilus Lee Khong Yoon

Submission Instructions

- You will need to submit the following files in your final project submission:
  - Your Jupyter Notebook report. Name the report `ProjectDesignReport<YourName>.ipynb`.
  - All relevant image files to be displayed in this report (make sure you use relative file referencing and the image will display in another co
  - Attached each file one by one and upload on Coursemology.
- Please print a copy of the final report to OneNote Individual Notebook space > Project. Double check on the image resolution. If the resolution is poor, please copy and paste the ORIGINAL clear image into the OneNote page (paste at the side of the printed image).
- Any submission that fails to comply to the above instructions will result in upto 5% penalty.
- You may wish to refer to the following reference to help organize and "beautify" your final report here.

https://thecodingbot.com/markdown-in-jupyter-ipython-notebook-cheatsheet/

Notebook link

Notebook link

Link to Notebook in hyperlink tag

Section A: Executive Summary

- Overview: A brief description of the project, its purpose, and the problem it addresses.
- Goals: Outline the key objectives of the database application.
- Target Users: Identify the primary stakeholders or users of the database.
- Justification for using RDBMS: Briefly justify why RDBMS is suitable in your case.

Overview

Advanced tools for tracking and analyzing user behavior on websites such as Sentry exist. However, we find that Sentry is more concerned with modelling the high level interactions of users with websites, for use in debugging. On the other hand, we believe that there is value in desiging a low level model of user browsing behaviour. Complex interactions between a user and a website cannot always be captured on a high level interpretation. For instance, Sentry's capability to interface with video controls has the following problems: a website front-end designer may decide that serving videos in a custom element is suitable. In that case, Sentry may face issues recording use of video control. However, a recording of only low level user input (click, drag, etc) will form a consistent pattern that is indenpendant of website implementation. Therefore, there are benefits to our approach over Sentry's.

We take inspiration from the work produced by this recent paper: https://huggingface.co/datasets/McGill-NLP/WebLINX. WebLINX has opened new doorways when it comes to training on browsing data, allowing AI models to mimic human browsing. However, the data collected is still on a small scale and is fairly complex to handle. A database management system designed solely for gathering user browsing data can streamline the process of data collection and inference. Hence, the database is useful in real life context.

Goals

- Allow users to record their browsing sessions in-system to be uploaded to the database.

- Allow researchers to conduct research efficiently.
- Allow database administrators the power to manage database.

Target Users

The database application's target audience are data science researchers, specifically, people specializing in web analytics or clickstream Analytics. UX Researchers may also find use in our database.

Data Science Researchers

- Easier to observe patterns in low level inputs than inputs that may have more complex meaning.

UX Researchers

- Useful for understanding how design tokens affect user interaction.

Machine Learning

- While we do not address directly, it is certainly possible to train models on our database.

Justification for using RDBMS

- On a high level understanding, the entities associated with a browsing session are highly related with each other. For example, a user click is inherently associated with a web page that the user is browsing on, and the html element that the user clicks on, and of course, the user click is associated with a user. This forms a rigid relationship between entities that we find is most efficiently represented by a relational schema.
- A Relational Database Management System favours scaling vertically (i.e. adding more server resources). This is ideal as our users consist mainly of individual researchers who are likely to have such a system, rather than a large audience of concurrent consumers accessing the database, in which case horizontal scaling would be preferred.
- A researcher would benefit from the greater ease of performing join queries in database management systems.
- A consistent schema is beneficial for tracking historical user data (i.e. we can still perform research across many years since the base schema does not change)

Section B: Business Rules

Complete your writeup of the business rules here.

When writing business rules for database design, ensure they describe how the system operates in real life, using clear and simple language. Avoid technical terms (e.g., "keys," "entities") and focus on real-world relationships and processes. The rules should easy enough for non-technical readers to understand, and yet specific and unambiguous enough for a database designer to model an EER diagram.

Good Example (Clear and Unambiguous):
Every student must be enrolled in at least one course. Each course can have many students, but a course cannot exist without at least one student enrolled. A student may also enroll in multiple courses.

Bad Example (Unclear and Technical):
Students are linked to courses with a many-to-many relationship. Foreign keys exist between tables for this purpose.

Some Considerations

- Position in DOM influences event execution order in the browser due to Event Bubbling and Event Capturing. We are not considering this for the database.

Users

Users are website accounts. Users contain an email address, user id, username and password. the email address is further split into email name and email domain. A User is capable of creating many Replays. A User may create no Replays, and every Replay must be created by a User.

Replay

Replays contain a snapshot of a recorded browsing session. UserEvents may take place in a Replay. A Replay may contain many UserEvents. A Replay can have no UserEvents. 2 Replays cannot contain the same UserEvent. Every Replay must contain exactly one of each of the following: unique replay id, time, browser agent, device type, device os, user ip address, default viewport. default viewport represents the dimensions of the viewport, and is further subdivided into width and height. IP address is further divided into network id and host id in accordance to ip address format. device Operating system is further split into os type and os version. Browser agent is the User-Agent header provided by the browser and is further subdivided into product, product version and comment, following user agent schema. time is split into the starting time

and the ending time. All time is measured in milliseconds elapsed since the UNIX epoch.

UserEvent
The database stores information about user events, where each UserEvent is an atomic browser action issued by a device due to user input. We will not be considering script-triggered events/inputs for this database. UserEvents are associated with Replays. Every UserEvent must have a Replay. Many UserEvents can be associated with the same Replay. UserEvents have a timestamp. Timestamps are measured in milliseconds elapsed since the UNIX epoch. Since it is reasonable to assume that a user cannot perform 2 inputs in the same millisecond, we can say that any 2 UserEvent timestamps are unique within a Replay. However, 2 UserEvent timestamps from 2 different Replays are not guaranteed to be unique. Each UserEvent also has 1 status determining the state of the action. The possible states are success, failure, timeout. Each UserEvent must contain a viewport width and height. A UserEvent results in a Webstate. Every UserEvent must have a Webstate. Only 1 Webstate can be associated with a UserEvent. There are many types of UserEvents. The types in consideration are 'NavigateEvent', 'ElementEvent'. Not every UserEvent must be one of the types, and a UserEvent may only have 1 type.

NavigateEvent
Some user events involve a change in route or domain of the web page, or in other words, a change in URL. We define these events to be called NavigateEvents. NavigateEvents are a type of UserEvent. NavigateEvents must contain at most 1 URL. Every NavigateEvent must have a URL.

ElementEvent
Some user events may involve 1 or more elements. We define these events to be called ElementEvents. ElementEvents are a type of UserEvent. ElementEvents have participating Elements. Not every ElementEvent has participating Elements. An ElementEvent may be associated with more than 1 Element. Many ElementEvents may be associated with the same Element. There are many types of ElementEvents. The types in consideration are 'TextEvent', 'MoveEvent', 'ClickEvent'. Not every ElementEvent must be one of the types, and an ElementEvent may only have 1 type.

TextEvent
TextEvent describes a change in text on a web page due to user or scripted input. TextEvents are a type of ElementEvent. TextEvents have new text to update the participating elements with. TextEvents must have a new text.

ClickEvent
ClickEvent describes a click input on a web page. ClickEvents are a type of ElementEvent. Each ClickEvent must contain x and y click coordinates. Each ClickEvent has a mouse button id. This corresponds to either the left, right, or middle mouse button. Each ClickEvent must have a mouse button id. Each ClickEvent must contain only 1 mouse button id.

MoveEvent
MoveEvent describes a movement of the cursor on the web page. MoveEvents are a type of ElementEvent. Each MoveEvent must contain exactly 1 x begin and y begin scroll coordinates. Each MoveEvent must contain exactly 1 x end and y end scroll coordinates. There are many types of MoveEvents. The types in consideration are 'ScrollEvent', 'DragEvent'. A MoveEvent may be either, none, or both of these types.

ScrollEvent
ScrollEvent describes a scroll input on a web page. ScrollEvents are a type of MoveEvent. ScrollEvents must contain the distance scrolled horizontally and vertically. This is stored as a fraction of the page width and height respectively. ScrollEvents may only scroll a participating element.

DragEvent
DragEvent describes a change in the position of the web page, with a mouse button in the down state. DragEvents are a type of MoveEvent. Each DragEvent begins and ends when the mouse button is pressed and released. DragEvents are a type of MoveEvent. Each DragEvent may only drag a participating element on the web page.

Element

Elements are entities that represent the elements on the website that the user interacts with. These entities are cherry picked heuristically to be relevant to user interaction. Elements may participate in ElementEvents. Many Elements can participate in many ElementEvents. Not every Element must participate in ElementEvents. Elements are also contained within a Webstate. Many Elements may be contained within a webstate. 2 Webstates may not contain the same element. An Element must be contained in a Webstate. Elements have uids. Any 2 Elements within a Webstate will not share uids. 2 Elements from different Webstates may share the same uid. An element has an element_id used as a key. An element must have 1 tag referring to its html element tag. All valid HTML tags are valid tags. Elements contain a xpath which identifies location of element in the page by traversing the DOM. XPath is needed in calculating tElements contain text which represent the text content in the html element. Elements contain an uid. Bounding boxes are further divided into x and y coordinates and a height and width. Elements may also contain many html attributes. Elements also contain a z_index, which identifies the layer on which the Elements exists on the web page.

Cookies
Cookies store website information used for authentication and user tracking among other uses. Cookies are associated with Webstates. Many Cookies may be associated with many Webstates. Cookies must be associated with a Webstate. Any 2 Webstates may contain the same Cookie. Cookies have a unique name. Cookies store whether they are secure, whether they are http_only and whether they are from the same site. Cookies store the following attributes: size, expiry, path, domain, value. Cookies store when they were last accessed.

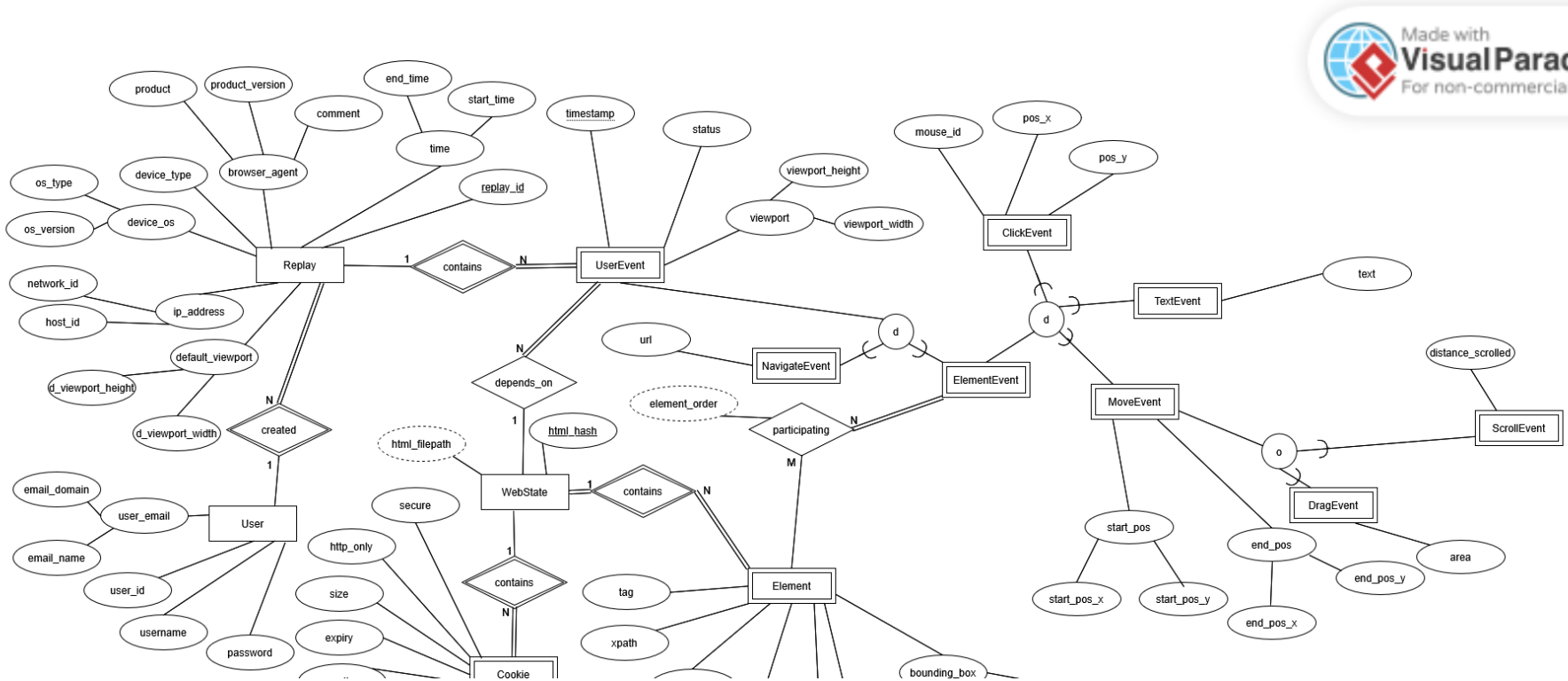
Webstate

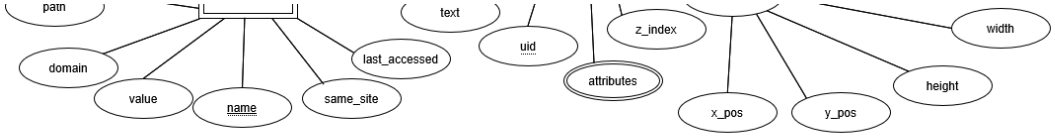
Webstates store minified html web pages. Webstates may contain Cookies. Many Webstates may contain many Cookies. Webstates may contain no Cookies. 2 Webstates may contain the same Cookie. Webstates are associated with a UserEvent. Not every Webstate must have a Userevent. Every Webstate must contain at most 1 UserEvent. 2 Webstates may not contain the same UserEvent. Webstates contain Elements. Each Webstate contains many elements. Each Webstate must contain at least 1 element. 2 Webstates may not contain the same element. A Webstate has a html hash. This stores the hash of the minified html of the web page. The hash is guaranteed to be unique. Webstates contain a html filepath, which locates the physical minified html file on a LFS system, and is composed of a base path concatenated to the html hash.

Participating Elements

Elements that participate in ElementEvents have an Element Order. The order value is calculated from the z_index and position within the DOM of each element, and then each element is sorted by the order value to get the element order.

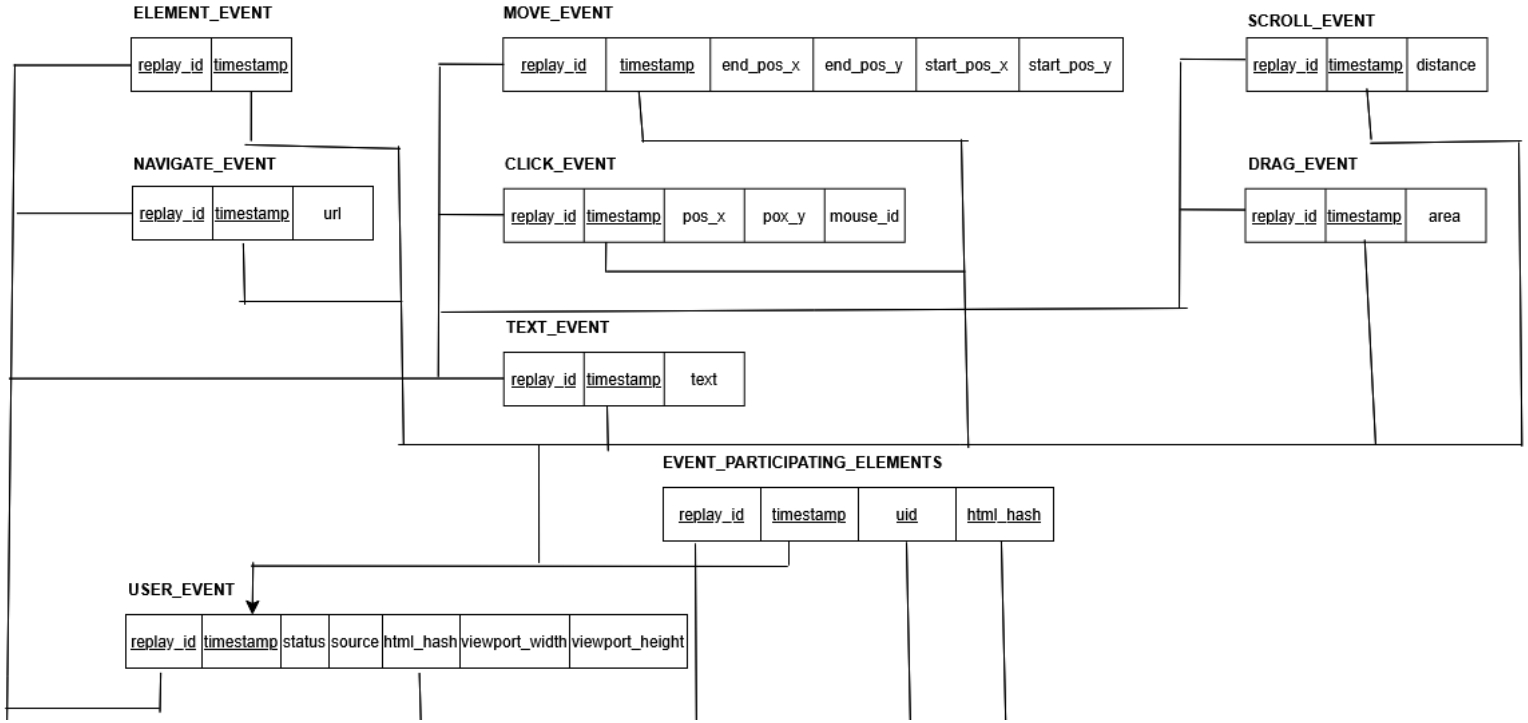
Section C: EER Model

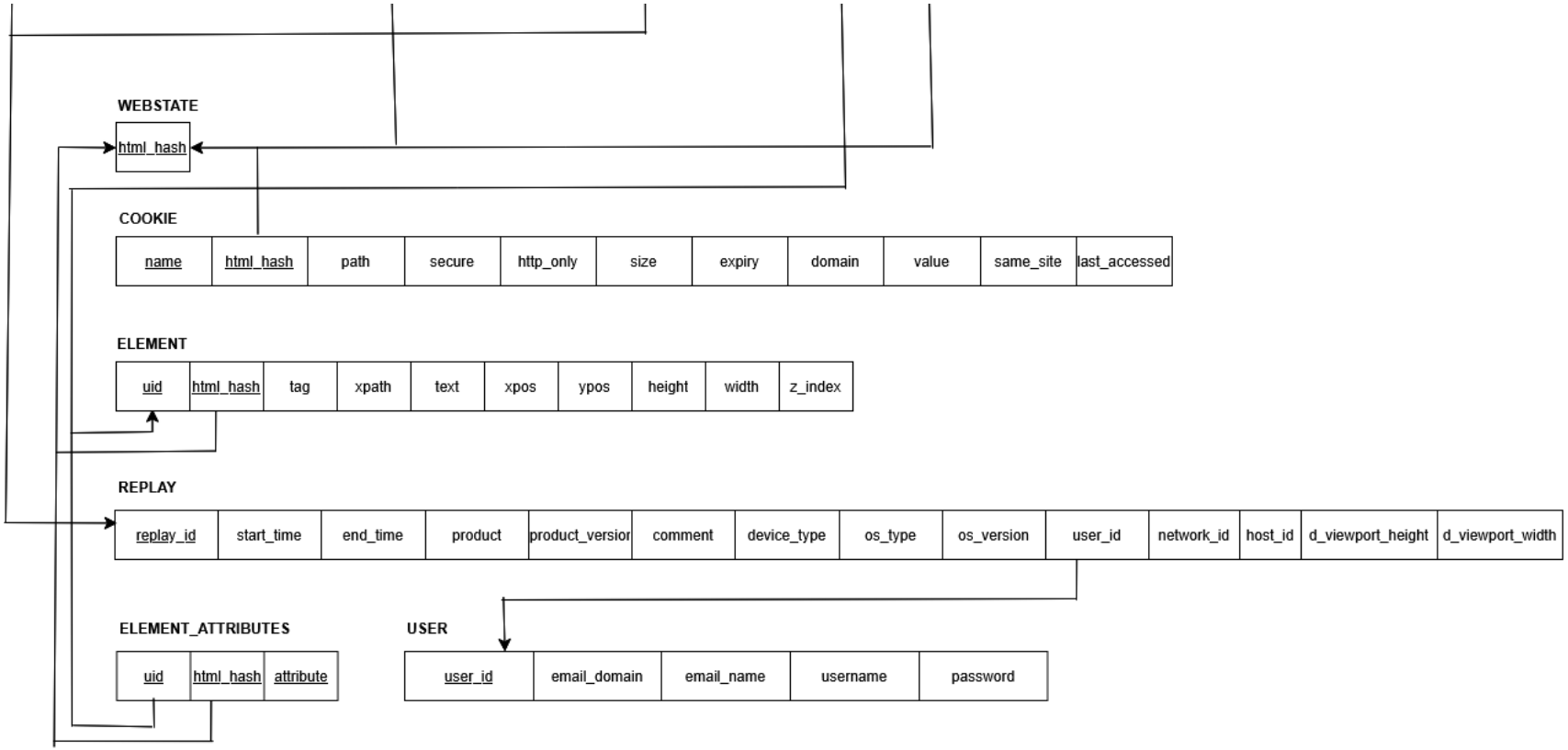




Section D: Relational Model

Attached the image of your Relational Model here. You should use Box-Style notation only. All foregin keys must be properly and NEATLY indica





Justify your mapping strategy from ER to relational, particularly if the approach deviates from the norm, or you have inheritance in your ER mod

Approaches that deviate from norm

There are some pecularities that may stand out in the mapping from ER to relational model, but they do not deviate from the norm. We will explain them here.

EventParticipatingElements Entity can be observed to have a composite key composing 4 key attributes. This is result of the following: ElementEvent and Element are related in a M to N relationship, with Element being a weak entity, and ElementEvent inheriting from a weak en
Therefore, we first use Mapping of Binary M:N Relationship Types to define EventParticipatingElements, containing weak keys timestamp and

Therefore, we first use Mapping of Binary M:N Relationship Types to define Event: participatingElements, containing weak keys timestamp and uid from ElementEvent and Element respectively. Then, we use Mapping of Weak Entity Types to include the respective strong keys, html_hash and replay_id.

There is also the high volume of weak entities present in the relational model and subsequently ER diagram. This is because in a webpage, most elements (cookies, html elements, events) have no meaning when taken out of their context (webpage). Therefore, it is necessary to represent this in the relational model. For instance, UserEvent is inherently dependent on context (Replay) to exist within the subject domain.

Specialization and Generalization
UserEvent is a superentity of NavigateEvent and ElementEvent. We chose the Multiple Relations-superclass and subclasses method for convertion to Relational Model. This is for the following reasons:
There are many possible input events, even on a low level interpretation, due to the wide range of input devices not covered here (pressure-sensitive touchpad, controller, etc.). Therefore, it is important to allow for ease of extension of the relational model. To achieve this, we must have a Generic UserEvent to offer minimal functionality for subclasses not defined in our relational model, which can then be updated to a specific subclass in the relational model when needed.
Also, it is important to note that events in Javascript are typically handled in a JSON-like format, and each only contain used attributes. Therefore, single relation with type attribute or 1 hot encoded attribute are not appropriate, since it will create unnecessary null values in the database, wasting space.

ElementEvent is a superentity of ClickEvent, TextEvent and MoveEvent. We chose the Multiple Relations-superclass and subclasses method. This is for the following reasons:
The same reasons from UserEvent will also apply here. Additionally, ElementEvent subclasses often contain position attributes, which may be confused with each other if they are combined into 1 relation.

MoveEvent is a superentity of DragEvent and ScrollEvent. We chose the Multiple Relations-superclass and subclasses method. This is for the following reasons:
The same reasons from UserEvent will also apply here.

If the relational schema mapped from the ER is not in 3NF, propose relevant normalization to make all relations in 3NF. You may leave this part b

Proposed normalization, if any

Section E: Schema Validation Using Conceptual User Needs

> Provide 2 conceptual user needs each of the 2 distinct user roles:

- Admin User: Define tasks or questions an admin user needs to perform or answer using the database (usually analytics in dashboard).
- Normal User: Define tasks or questions a typical user (e.g., a customer, student, or employee) needs to perform or answer using the database

For each need:

- Clearly describe the user requirement in plain, everyday language.
- List the entities and relationships in the EER model that are involved in supporting this need.
- Describe how the EER model supports the user need, focusing on how entities are connected.

For example:

1. User Need: Administrators should be able to view a list of all courses and the number of students enrolled in each course.
2. Entities and Relationships Involved:

- Entities: Course, Student, Enrollment.
- Relationships: Course is related to Enrollment. Student is related to Enrollment.

3. EER Model Support: The Enrollment relationship records which students are enrolled in which courses. By counting the number of entries in the Enrollment relationship for a specific course, the system can determine how many students are enrolled.

User

User Need: Users should be able to review their own recordings.

Entities and Relationships Involved:

1. Entities: User, Replay

2. Relationships: User is related to Replay

EER Model: The database is able to only retrieve recordings associated with a specific user.

User Need: Users are able to query how UserEvents affect the Webstates for AI training.

1. Entities: UserEvent, Webstate
2. Relationships: UserEvent is related to Webstate

EER Model: Webstates related to a specfic UserEvent can be found in the database due to their relation.

Admin User

User Need: Adminstrators should be able to remove everything belonging to a specific recording.

Entities and Relationships Involved:

1. Entities: Replay, UserEvent, Webstate
2. Relationships: Webstates depend on UserEvents which in turn are contained by Replay

EER Model: Everthing involving a specific Replay can be deleted together easily by checking relationships. Deletion can be easily done by following dependencies.

User Need: Administrators should be able to montior the interactions of specific elements.

1. Entities: ElementEvent, Elements
2. Relationships: Elements are related to ElementEvents

EER Model: There is a relationship between Elements and ElementEvents so the system can determine how which ElementEvents work on a specfic Element.

Contribution Log

Clearly state the contribution of each member below:

Ang Wei Bin

Task/Module	Contribution Details
Buisness Rules	Planning, Replay, Element
ER Model Design	Replay, Element, revised some elements
Relational Schema Design	Reviewed the design
Schema Validation Using Conceptual User Needs	Entire Schema Validation

Theophilus Lee Khong Yoon

Task/Module	Contribution Details
Executive Summary	Entire executive summary
Business Rules	UserEvent, planning, refine business rules to match final ER Diagram
ER Diagram	User, UserEvent children, Element Webstate Cookie
Relational Schema Design	Converted ER model to schema, wrote justifications
Schema Validation using conceptual user needs	-

Citations

Deepnote's text autocompletion feature (.powered by codium) was used for writing this document. Deepseek R1 and Cursor Tab were also used for autocompletion on the Business Rules. ChatGPT was used for ideation and research.

ChatGPT Logs

- https://chatgpt.com/share/67a87226-b150-8000-8717-41dc5aca7a2e
- https://chatgpt.com/share/67a87245-7ff4-8000-af5a-8f63591a5159

- https://chatgpt.com/share/67a87255-c54c-8000-a0f1-89209bc8481e
- https://chatgpt.com/share/67a8726f-fac4-8000-a6f8-309bdc94381b
- https://chatgpt.com/share/67a8732b-3ac8-8002-a733-e4c97b819bc3