A Collaborative Visual Database

by

Imed Adel

A document submitted in partial fulfillment of the requirements for the degree of

Technical Report

at

Miskatonic University

Contents

List of Figures

List of Tables

General Introduction

Software is either slow, hard, or ugly. And sometimes all three. Nonetheless, since the introduction of the modern computer, software has taken the world by storm. And it quickly became an essential component of every business since it paved the way for higher productivity and more automation, and therefore, increased profits.

Nearly forty years [1] have passed since the launch of the Apple Macintosh—one of the first commercially successful [2] mass-produced personal computers featuring a graphical user interface. Yet, software is still as inaccessible and inadequate for most users as ever. Perhaps the best example for such inaccessibility is the fact that this document is being produced using LaTeX—a fractured software system that requires a plethora of tools to be installed for the sake of producing a legible and aesthetically pleasing document.

Along with the domination of personal computers and software companies in the world, another technology was on the rise—the internet. Since the dotcom bubble in the early 2000s, the internet has reshaped our lives. Be it entertainment, communication, education, or work, the internet is the primary and most powerful medium. Therefore, it is no surprise that most software companies switched to Software as a Service (SaaS), that is hosted software served through the medium of the internet [3]. Which quickly evolved into collaborative software aimed at teams rather than individual users.

The continued sprawl of these technological advancements led to the rise of remote work—a movement that erases any geographical limits and allows businesses and institutions to expand well beyond their headquarters. This movement has been recently magnified to unprecedented levels due to the global pandemic. The main traits of work and education instantly changed and non-collaborative software fell behind to give room to collaborative SaaS.

Within these changing dynamics, managing data is still an unsolved problem. Setting, managing, and securing a database is still one of the hardest tasks of building a business. Connecting the database to the rest of the business' applications is not as easy as one might expect. Keeping all employees on-board and managing access to the database, while allowing everyone to seamlessly collaborate is not easily achievable. Requiring all of the above while keeping the costs low is impossible. Software geard towards managing data and content is either hard to configure and hard to use or slow to load and slow to on-board.

Based on the belief that software must be accessible, collaborative, fast, and hopefully enjoyable to use, we set out to develop a modern alternative. Using the latest technological innovations, our project is pushing the limits for what is possible with collaborative SaaS for managing data and content. Merebase—our project—is a collaborative visual database SaaS that challenges the norms, democratizes access to data management software, and fills the need for a no-code and low-cost database software.

Our work is discussed in four chapters.

- Presentation is an introduction.
- Analysis and specification of needs is an analysis.

I Presentation

1.1 Introduction

The aim of this chapter is to contextualize our work. We will start by introducing the hosting institution for the graduation project. Then, we will present the project, its motivations, and its objectives. And finally, we will discuss the development process used throughout the making of this project.

1.2 Host

This project is done as part of the final graduation project with the goal of obtaining the License Degree in Computer Science within the Higher School of Sciences and Technology of Hammam Sousse.

1.3 Project presentation

Our project's main idea and design choices stem from the problems we faced while trying to accomplish certain tasks using other tools.

1.3.1 Problematics

The continous shift to Software as a Service (SaaS), coupled with the rise of remote work, uncovered a gap in the field of data and content management software. The gap is further exacerbated due to the accelarting adoption of web applications, which are mostly client-side applications without any server requirements. Nowadays, businesses are looking for easy and collaborative ways to allow stakeholders to manage data and content, and to connect the data to their different applications. The solution must respond to the needs of businesses from different backgrounds, with varying budgets, and minimal technical knowledge. The solution must also be easily intergratable with other tools that these businesses might rely on. Furthermore, the solution must support recent technilogical advancements in the web, such as real-time collbaoration and real-time queries. To ensure these requirements, we need to answer the following questions:

- How to support real-time collaboration?
- What level of collaboration is required for optimal productivity?
- How should we organize and share data between multiple users?
- What interface structure ensures the most accesible software?
- What data types should we support?
- How important is speed?
- How can we ensure a fast user experience?
- What are the bottlenecks of the existing solutions and how can we solve them?
- How can we ensure a fast and easy on-boarding?

1.4 Preliminary study

Before starting the development process of our projects, it is of utmost importance to research the existing solutions in the field of data and content management that our potential users are currently relying on. It is necessary to understand what problems users are facing while using these solutions and what kind of tricks and shortcuts do they have to depend on to achieve their desired outcome. We should also focus on the points that users admire about their current choices, as these are the features keeping them from looking for another solution in the meantime.

With this goal in mind, we went on to research multiple applications and software systems with varying degrees of features and requirements. While some might require deep technical knowledge of databases, servers, and programming, others are more straightforward and require little to no technical knowledge. However, while some applications may require no programming skills, they still require some time for on-boarding and getting familiar with the software. This can be a significant roadblock for many entreprises that are already stuck with some other software system.

From this wide pool of data and content management software, we selected the most used and loved ones and put them to comparison. In particular, we chose to focus on PostgREST, Notion, Airtable, Contentful, Sanity, Webflow CMS, and Firebase.

1.4.1 Existing solutions

We will start by presenting the selected solutions.

Figure 1.1: PostgREST logo

PostgREST

PostgREST is an automatic API generator for Postgres databases.

Firebase

Figure 1.2: Firebase logo

Firebase is a platform developed by Google for creating mobile and web applications. It was initially released in 2012. It offers, among its products, a real-time database. In which, data is stored in JSON format and synced between all the connected clients. The database was not developed with non-technical users in mind, however, its real-time capabilities offer an example of what's desired in real-time database software. Firebase Realtime Database has been successfully used to develop highly demanding mobile applications.

Airtable

Figure 1.3: Airtable logo

Airtable is a visual database app inspired by the ease of spreadsheets and the wide adoption of software like Microsoft Excel. The company behind the app was founded in 2012.

Airtable comes with team collaboration out of the box. It also automatically generates a REST API from each database.

Pricing is done per team member. There are several limits to the size of storage and uploads.

Contentful

Figure 1.4: Contentful logo

Contentful is a headless[1] CMS (Content Management Software). It offers a flexible CMS editor and a configurable API. It also comes with multiple SDKs

[1] Content is decoupled from the main application. It's made accessible through a set of APIs.

(Software Development Kits) in multiple programming languages to make its integration easier.

Pricing is offered per package, with the lowest premium package starting at US$489 per month.

Sanity

Figure 1.5: Sanity logo

Sanity is another headless CMS. It competes directly with Contentful, offers an even more configurable editor, and its pricing starts at US$199 per month. It comes with real-time collaboration, a feature that Contentful lacks.

Webflow CMS

Figure 1.6: Webflow logo

Webflow is a website builder. It bundles a CMS and an e-commerce management system along with its visual website builder. The CMS is not usable outside of Webflow websites, however, it comes with an intuitive user interface.

Notion

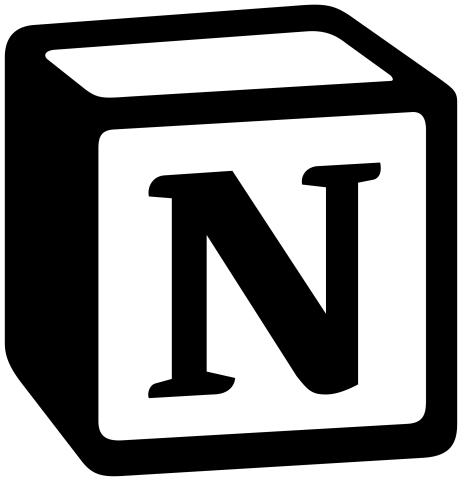

Figure 1.7: Notion logo

Notion is a new contender in the space of content management. It presents itself as a collaborative workspace for teams. Its use cases vary from product management and team documentation to note-taking and personal organization. The initial version of Notion was released in 2016. The second version, which received a lot of praise and media coverage, was released two years later in 2018. However, the largest surge in signups happened during the pandemic, with 40% of signups occurring from December 2020 to January.

Notion is built on the concept of blocks: A block is any single piece of content you add to your page, like a to-do item, an image, a code block, an embedded file, etc. [2] This makes it easy to build complex pages and move content around.

Notion is also built as a collaborative web app—eliminating the need for saving and figuring out how to share one's documents as is the case in other apps.

[2] citation needed, see Notion FAQ

Pricing is done per workspace member with unlimited storage starting from the free plan.

1.4.2 Comparison of the existing solutions

In order to have a better understanding of the different offerings of the selected solutions, and their features and shortcomings, we have to compare them side by side.

Methodology

For a fair and objective comparison of the different solutions, we will rely on Web.Dev and Google Chrome Lighthouse for measuring performance and accesibility speed, and Capterra for aggregating user reviews.

Web.Dev Web.Dev is a tool developed by Google that uses the Lighthouse enginge to measure different websites and web applications metrics.

Google Chrome Lighthouse Web.Dev is a tool developed by Google that uses the Lighthouse enginge to measure different websites and web applications metrics.

Capterra Web.Dev is a tool developed by Google that uses the Lighthouse enginge to measure different websites and web applications metrics.

Comparison table

Table 1.1 is an objective side-by-side comparison of our selected solutions.

	PostgREST	Firebase	Airtable	Contentful	Sanity	Webflow CMS	Notion
Released	2016	2016	2012	2016	2016	2012	2012
Type	Databse / API	Database	Spreadsheet / database	CMS	CMS	CMS	Note-taking
Requires	Yes	Yes	No	No	No	No	No
API	Yes	Yes	Yes	Yes	Yes	Partial	No
4	545	18744	7560	55	44	55	44
5	88	788	6344	Contentful	Sanity	Notion	Njj

Table 1.1: Comparative table of the existing solutions

1.4.3 Critique

Multiple solutions are trying to focus on various use cases, however, all of them suffer from noticeable performance issues, a bad UX (User Experience), and inadequate pricing for small and medium-sized businesses.

Notion is known for its slow performance and long loading times. Pages take on average between six and 12 seconds to load. [3] It also doesn't have an API, although one is being developed at the time of writing. Furthermore, Notion is less structured than products like Airtable or Firebase.

Airtable is notable for its complexity, even for experienced users. It also suffers from some performance issues when loading large documents. Furthermore, it doesn't have the same rich text capabilities as Notion. Finally, it lacks a real-time API and it's relatively expensive.

Case study: Notion

Notion is great.

[3] citation needed

1.4.4 Proposed solution

Merebase is a collaborative visual database that can be used for data and content management. It's built with real-time collaboration, performance, and intuitiveness in mind. Thanks to years of innovation in the field of browser apps and high-performance real-time servers, it should be able to load instantaneously, while offering a smooth user experience with no glitching or slowdowns when loading large documents, and with the ability to effortlessly collaborate with other users.

1.5 Development process

To ensure the optimal use of time and energy, we chose to follow a development process throughout this project. That is, dividing work into smaller chunks according to a certain set of rules. In particular, we followed the principles of Agile software development, which is an umbrella for multiple methodologies and frameworks. Of these methodologies, we adopted Feature-Driven Development (FDD) and the Kanban method for their practicality and ease-of-use, especially for projects with small teams.

1.5.1 Agile software development

Agile software development is an umbrella term for a set of frameworks and practices based on the set of principles popularized by the Manifesto for Agile Software Development in 2001. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change. It derives its values from a range of software development frameworks and methodologies.

Values

The Manifesto for Agile Software Development proclaims the following values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Principles

The Manifesto for Agile Software Development is based on twelve principles:

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams

12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

1.5.2 Feature-Driven Development

Feature-Driven Development (FDD) is an Agile method for developing software iteratively and incrementally. It encourages planning, design, and development based on features.

1.5.3 Kanban

Kanban is an Agile method to manage work by balancing demands with available capacity, while uncovering bottlenecks. Work is devided into smaller tasks that are visualized on top of a Kanban board.

1.6 Conclusion

In summary, throughout this chapter, we have presented our project, the hosting institution, the motivations behind our choices, and our objectives. We also researched the existing solutions and we started drawing the picture for what our project strives to achieve.

Whithin the next chapter, we will be going into more details of this picture by focusing on the analysis and specification of needs for out project.

2 Analysis and specification of needs

Researching the current solutions led us to formulate a set of requirements to ensure that Merebase offers the best experience.

2.1 Functional requirements

- A user must signup and login using only their email
- A user's account picture is fetched automatically from Gravatar
- A user can create a maximum of 20 workspaces [1]
- A user can invite other users to their workspace using their email
- A user can create new projects, columns, and rows
- A user can query the database using a REST endpoint and a Websocket endpoint
- A user can upgrade their account to a premium one
- A user can cancel their premium subscription
- A user can edit the same document as other users at the same time

[1] This is a technical limit imposed by Stripe, the payment processor

- A user can define the column data type (text, number, boolean, etc.)

2.2 Non-functional requirements

- The web app should load within milliseconds
- Browsing large documents should not result in glitches or lags
- The interface should be accessible and intuitive
- Private documents should remain private and inaccessible to hackers
- The web app and the real-time server should be always available

2.3 Identification of actors

Merebase uses RBAC (Role-Based Access Control) to manage users' access levels and permissions. There is only one actor, the user, but with multiple assignable roles.

- Owner: The user who created the resource, be it the workspace or the project. This role gives you entire access to the resource and it is assigned automatically.
- Admin: This role gives non-owner users the same privileges as the owner. Admins can invite new users and assign roles.
- Editor: This role permits a user to edit documents in a workspace.
- Viewer: This role permits a user to view documents in a workspace, without the ability to modify them.

2.4 Use case diagrams

To better illustrate the main interactions between the user and the application, we rely on a use case diagram.

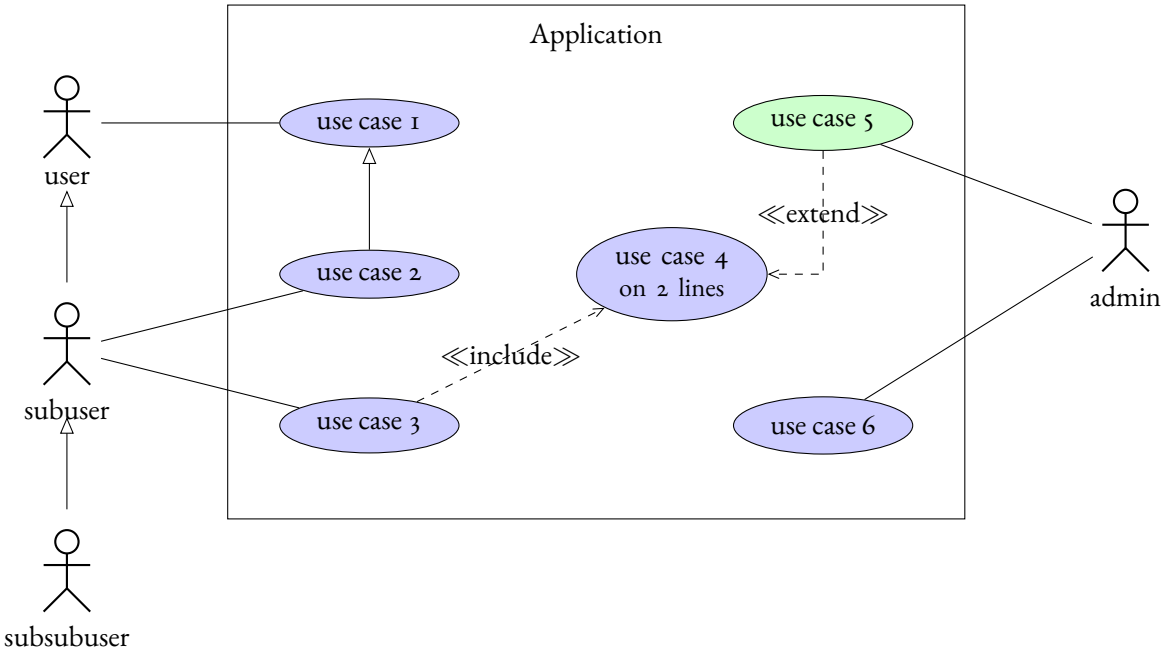

Figure 2.1: Example of a parametric plot $(\sin(x), \cos(x), x)$

2.5 Conclusion

TBD

Acronyms

CMS Content Management Software

FDD Feature-Driven Development

SaaS Software as a Service

Bibliography

[1] *Macintosh Product Introduction Plan*. July 2010. URL: https://web.archive.org/web/20100721013724/http://library.stanford.edu/mac/primary/docs/pip83.html (visited on 04/01/2021).

[2] K. Polsson. *Chronology of Apple Computer Personal Computers (1984-1985)*. Aug. 2009. URL: https://web.archive.org/web/20090821105822/http://www.islandnet.com/~kpolsson/applehis/appl1984.htm (visited on 04/01/2021).

[3] B. Turner. *What is SaaS? Everything you need to know about Software as a Service*. en. URL: https://www.techradar.com/news/what-is-saas (visited on 04/01/2021).