BDM - First Deliverable

By

Kristóf Balázs
Josu Bernal
Stefanos Kypritidis

Professors:
Besim Bilalli
Achraf Hmimou Ham Man
Marc Maynou
Uchechukwu Fortune Njoku
Sergi Nadal

Barcelona, April 2025

Project context

After a lot of iterations in the courses of viability of business projects and big data management, we finally settled with the domains of networking and dating offering a unique value proposition to our future customers. The struggle before was to find a solution/project idea that would be unique, interesting, promising and at the same time fulfill the requirements of both the mentioned courses. Something noteworthy to mention is that with the rise of AI and LLMs most innovative ideas we thought about were already implemented, highlighting the difficulty of pioneering in the present business world.

Our final idea is the app "Lets Talk" which aims to help users connect meaningfully by generating dynamic, up-to-date questions based on shared interests. Designed for individuals seeking to quickly discover common interests and spark engaging conversations, it is an app that instantly compares the profiles of two users and generates fascinating questions. In the final stage, we visualized that this connection is implemented using NFC or qr-codes to reduce manual tasks and automate the process for the users. This new method will make it easier to find common ground, making social interactions feel more fun.

Domain

The main domain of our application is dating while it can be extended to networking and communication in general. Our application can be a great tool for singles trying to create impactful relationships by bypassing the awkward ice-breaking phase and quickly diving into the more engaging and authentic connection-building stage. Furthermore, our application can be an excellent solution for networking environments e.g. business networking events or student events, where it is often challenging to move past small talk and engage in more meaningful, stimulating conversations that benefit both parties. In short, the app merges technology, social psychology, and user engagement.

Of course, various other industries can be targeted with our application. In detail, the app could be a great tool for travelers in the tourism industry, assisting them to form meaningful connections in settings like hostels or travel groups. Moreover, the app can be adopted by HR departments of companies to ease the onboarding of new employees and enhance employee engagement, team building, internal networking or collaborative culture development.

Some key characteristics of the application include:

- Connecting with a nearby user which produces dynamic, up-to-date questions based on the users' interests. The questions aim to create enthralling conversations on topics that both users enjoy. The connection can be later implemented by scanning a qr-code or by NFC technology to make it a seamless and effortless experience for the user.
- User dashboard that contains relevant news and information that may be useful or interesting

Business Value

Despite social media aiming to connect us and making us more social, it turns out that it makes us unsocial and isolated. It is in fact considered as a modern addiction Jafari (2025). In addition, the users belonging to the age groups of 16-24 and 25-34 are the most active on social media and thus the most at risk of facing the consequences Social (2024). Nowadays, it's common to see young people spending time together while glued to their phones, exchanging only a few words and rarely engaging in meaningful conversations. Not only has that transformed and limited interpersonal interactions the recent years, but it has also deteriorated the social skills of young people. As a result, oftentimes it is very hard for young people to start interesting and deep conversations that can lead to new friendships, collaborations, or romantic relationships.

To sum up the problems, many people struggle with starting meaningful conversations, often leading to awkward interactions and shallow discussions due to a lack of common interesting topics. Traditional icebreaker methods, such as pre-set questions, are often too generic and fail to cater to individual interests, making conversations feel forced or unengaging. Additionally, manually discovering shared interests can be time-consuming and inefficient, especially in a fast-paced environment like networking events or speed dating events, where quick and meaningful connections are preferred.

Our idea tries to tackle those issues by utilizing technology, which is an integral part of our lives and especially of the young people's lives. "Let's talk" is an app that is made for people to connect meaningfully during in-person meetings. Two users, who have just met, can quickly connect in the app, and then a plethora of invigorating questions will be produced. This helps pass the "break the ice" phase and avoid awkward silences while the two people do not know what to talk about. The app could even be gamified in a way that the two users have extra motive to stay engaged and answer more questions. In conclusion, a trend like this could capture the attention of the younger generation, sparking change toward a more interpersonal and less screen-dominated future.

In essence, our solution leverages Big Data technologies and smart question generation by analyzing user profiles, interests, and preferences to create personalized and engaging conversation starters. To make connecting easier, users can instantly connect via NFC or by scanning QR codes, eliminating the need for manual profile searches and making the experience seamless. Last but not least, the app ensures up-to-date and contextual questions by dynamically updating topics based on real-time trends, keeping conversations new, meaningful, and enjoyable.

There are various potential competitors. On the one side, dating apps like *Bumble, Tinder, and Hinge* prioritize connecting people rather than facilitating deep, engaging conversations. Also, *TableTopics* offers decks of questions and an app with fun icebreaker prompts but lacks a focus on meaningful dialogue. Moreover, *Lunchclub* connects users based on shared interests and goals through AI but does not provide question generation. In addition, *Party Qs* is an app

for icebreaking and self-discovery, offering static, non-personalized questions based on chosen topics, while the app *KnowMeBetter* presents random questions to help people learn about each other. Additionally, *Let's Get Deep* is an app that fosters genuine connections with pre-programmed questions. Lastly, *The And (by The Skin Deep)* provides physical card decks with thought-provoking questions for different relationships, and *Actually* Curious" offers predefined card-based question prompts to encourage meaningful conversations.

Approach and technology

In this section, we will explore in detail how we modeled and approached the first part of the project, which focuses on the landing zone. This section will be divided into three subsections: first, we will discuss the data sources we will be using; second, we will cover the technology employed; and finally, we will address the relevant reasons behind our choices and the methods we used.

Before we begin, it's important to clarify that our business idea is quite broad and ambitious. To ensure we have a clear focus for our project, we decided to limit our scope from the outset. For now, both teams will only work with data related to the following topics: general questions or app capabilities, sports, video, and technology. The idea, if the business wants to grow, is to add more topics to make conversations increasingly personalized and engaging.

Data Sources

In this first subsection, we will only describe the sources, we won't explain how or what we did with them. We think it is important to do this separately because it makes the problem easier to understand. The data source search process was conducted in collaboration with the other subteam. After the search was performed, we divided the sources based on each subteam member's needs and capabilities. Ultimately, our subteam worked with five distinct data sources, ensuring they covered a variety of technologies relevant to the project.

1. **Users database:** This is a structured streaming synthetic database designed to simulate the data that users will store in their profiles. This is not the operational database under the app itself but a weekly backup used for analytics. With this data source, we aim to demonstrate our capability of generating synthetic data while working with both relational and streaming databases. The database is not large in volume, as it is unnecessary to use a significant workload to showcase our ability to manage it. Additionally, it is limited in terms of features, but it plays a crucial role in defining the scope of the project. As previously mentioned, we will store only generic personal information about users and their interests in three topics: sports, video, and technology.

2. **Reddit API:** Reddit is a social media platform where users can share and discuss content in groups called subreddits. We used its API to get hot or trendy conversation topics in different areas, which were easy to filter based on the subreddit they where posted on. We selected a total of 14 subreddits for our use case. This API stores streaming semi-structured data. With this, we wanted to show our capability of working with APIs, semi-structured and streaming data.

3. **Stack Exchange API:** Stack Exchange is a platform of expert communities to share knowledge on focused topics. It has a total of 173 communities. We used its API, that stores

semi-structured data, which fits our technology topic needs. With this data source, we wanted to show our capability of working with semi-structured data.

4. **Trivia questions API:** This is an API that stores semi-structured data of multiple topics. It is fun and less intense than the others, and we thought it could be a great addition to make the conversations have some kind of playfulness and allow some learning. With this, we wanted to show our capability of working with APIs and semi-structured data.

5. **Wikipedia API:** Wikipedia is a free online encyclopedia that is written and maintained by volunteers. Their API stores unstructured articles. With this data source, we wanted to show our capability of working with unstructured data.

Technology

After studying our options and discussing the pros and cons with the professor, we decided to create a data lake as our landing zone. We had two main options: we could either set it up on our local desktop using technologies such as Delta Lake, an open-source data lake platform, or utilize a cloud platform. Ultimately, we chose to go with the cloud option because its scalability, performance, and integration advantages. We also thought that it could be a great opportunity to learn how to use this technology.

After comparing the capabilities offered in the free tiers of various cloud platforms and considering our limited prior experience, we decided to proceed with Google Cloud, as it was the most cost-effective choice.

Google offers many services as part of their platform, but in this case, we used the following:

- **Cloud Storage:** Google Cloud's storage solution that allows you to create different types of buckets based on the required read and write concurrency. You can store various types of data within these buckets. For this purpose, we will use a Standard Storage bucket as a data lake, which we will divide into folders to store the different data sources.

- **Cloud Run Functions:** This service enables defining functions in various programming languages and executing them on command, triggered by events or as scheduled tasks. This will be crucial to ingest streaming data.

- **Cloud Scheduler:** This service allows you to schedule different cloud services, such as running predefined cloud run functions. This capability will allow us to schedule the streaming ingestion.

Let's review how we worked with the various data sources.

1. **Users database:** For this data source, we began by defining the information we wanted to store and designing the appropriate schema. The biggest challenge was ensuring that we could work simultaneously with the operational database supporting the app and the data lake, keeping everything connected. To achieve this, we created synthetic data using a Python script (in a Jupyter notebook) and uploaded it to Firebase, the technology we chose for the operational app database. We selected Firebase because it is a Google product, which helps with compatibility issues and simplifies the process.

   Firebase stores data in a semi-structured format, using the column ID as an identifier for a JSON object that contains the rest of the columns. It is also highly exportable as a structured table, compatible with Big Query, which was one of the reasons we opted for this technology. After the data was uploaded to Firebase, and we tested it in the app prototype (developed for a business class), we defined a Google Cloud Run function and scheduler to export this information to Google Cloud Storage.

   It's important to note that we will only export the tables relevant for analytics; we will not include operational information or sensitive data such as passwords or emails. Finally, we scheduled the function to run every Sunday at midnight, as we believe this is a suitable time to have the new data ready by Monday morning for analysis. We do not think a higher frequency of streaming is necessary since we are not planning to launch the app yet, which means the database will not change significantly. However, we wanted to demonstrate our capability of working with streaming data.

2. **Reddit API:** We use the dedicated Reddit API (without any wrapper like PRAW), with a script that first requests an access token to bypass the standard rate limit (10 requests per minute) and instead allows for 100 requests per minute. It then goes over a predefined list of subreddits, which we selected manually by researching each (14 in total) and it fetches up to 1,000 top posts by upvote count per subreddit. For each subreddit, the script sends requests in batches (100 posts per request) using pagination. It is worth mentioning that the code gets the raw JSON with every field available and no filtering is done as the API is subject to change. Historical data is uploaded to a Cloud Storage bucket using service account credentials in the Google Cloud SDK.

   Besides getting historical data, we get the most upvoted questions (top 10 in each subreddit) in the last 24 hours for each day as these usually have questions on recent/hot topics. For this, we use a containerized Python application which we deployed in Google Cloud Run and trigger each day with Cloud Scheduler. Data is arriving to a subfolder with the date of fetching. This is our main streaming data source.

3. **Stack Exchange API:** In this case, we also use the dedicated API with simple GET requests for the raw JSON data. We selected a total of 10 communities which are relevant for our proof of concept and we get the 1000 most upvoted posts in each community. As

we ran into API limits often here, the fetcher Python code is designed in such a way to bypass these where possible. As with the case of other historical sources, it is uploaded to a Cloud Storage bucket using service account credentials in the Google Cloud SDK.

4. **Trivia questions API:**

   We created a script to collect and store trivia questions using an online trivia API and Google Cloud Storage (GCS). The script fetches trivia questions from Open Trivia DB, which limits responses to 50 questions per request. To get more than 50 questions-data, the script runs a loop of multiple requests, waiting a few seconds between each request to avoid overwhelming the API server. In order to avoid getting duplicate questions, each question text is saved in a Python set, and a check is made to ensure that the new question has not been retrieved before.

   Once all the trivia questions are collected, they are saved into a single JSON file on the local system. Afterward, the file is uploaded to Google Cloud Storage. The file is stored in a specific subfolder of the main bucket, making it easy to access later. This process ensures that we gather a large set of trivia questions (around 4,500).

5. **Wikipedia API:** A script was created to collect and store Wikipedia page content related to specific topics using the Wikipedia API and Google Cloud Storage, respectively. The script first searches Wikipedia for pages matching a given topic (in our case, sport, film, and tech), retrieving up to 1000 search results per topic. For each result, the script then fetches the full page content in plain text format using the API.

   After gathering all the pages related to a specific topic, they are stored in a JSON file within a local directory. The json file contains a list of dictionaries, each one having the keys 'title' and 'content'. The file is named according to the topic (e.g., sport_pages.json for Sport). Once the files are saved, they are uploaded to Google Cloud Storage. The files are placed in a specific subfolder inside the main bucket, keeping them well-organized and easy to retrieve.

The clear diagram of the current data pipeline can be seen below. To sum it up, this data pipeline integrates structured, semi-structured, and unstructured data sources into a Google Cloud-based landing zone. Users' data is synthetically created via a Jupyter notebook and loaded into Firebase. The new user data is retrieved from a JavaScript script triggered by Google Cloud Scheduler. Similarly, the top-10 daily questions per subreddit are retrieved from a containerized python script in Cloud Run, which is triggered by a Google Cloud Scheduler for streaming data ingestion. Moreover, semi-structured data from trivia questions, Reddit, and Stack Exchange and unstructured data from Wikipedia are accessed via Python scripts API calls. The processed data, including user data and JSON-formatted trivia questions, Reddit, Stack Exchange, and Wikipedia content, is stored in the Google Cloud landing zone.

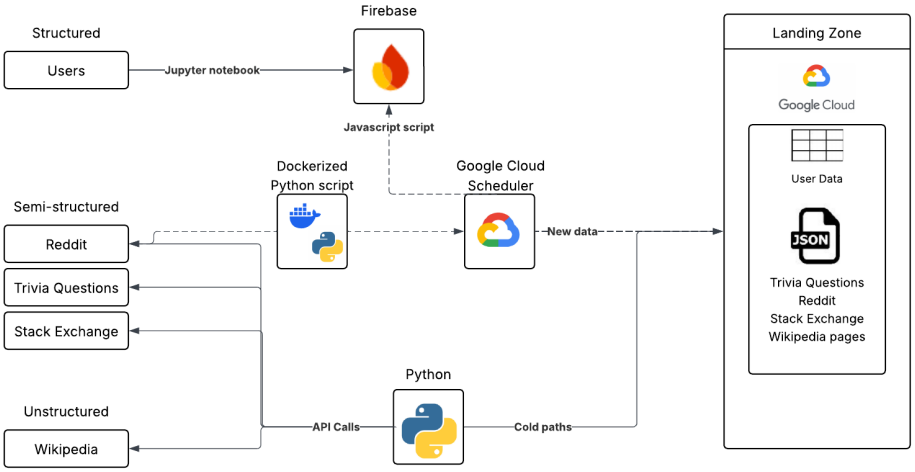

Figure 1: Data Pipeline

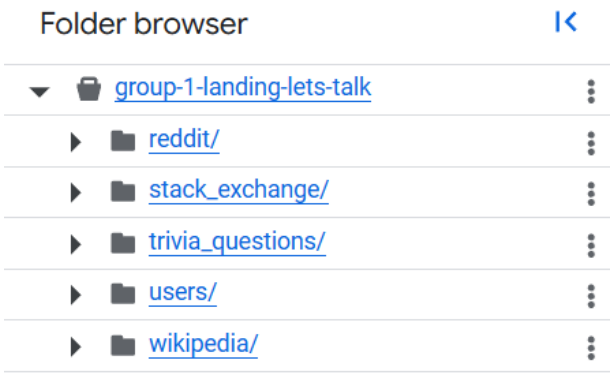

Figure 2: File Structure in Google Cloud Storage

Relevant choices

The process and decisions we discussed in the previous paragraphs were not made instantly or without thought. We underwent an extensive trial-and-error process to understand the true benefits and characteristics of various approaches. For instance, we tried using Google Cloud Composer, Google's orchestration tool, which resulted in significant charges against our free credits. In hindsight, it turned out to be excessive for what a simple scheduler could easily accomplish. We also tried Apache Airflow, using cloud command line, but we ended up choosing what we think it is the best and most affordable option (which business wise is also important).

References

R. Jafari. Unsocial media: How social media is isolating us. https://medium.com/@ranajafariii/unsocial-media-how-social-media-is-isolating-us-34570cbda2e5, 2025. Accessed: 2025-03-24.

S. Social. New social media demographics, 2024. URL https://sproutsocial.com/insights/new-social-media-demographics/#:~:text=18%2D29%20years%20%E2%80%93%20YouTube%20(,22%25)%2C%20Instagram%20(19%25). Accessed: 2025-03-30.