MVP PRD: Proxima

2024.04.20

Overview

> *Imagine a social network where every post becomes a destination, encouraging you to step out and explore. It is not just about scrolling through life — it is about living it, one post at a time.*

Proxima transforms the way people connect online by anchoring social interactions in real-world locations. Users can create and share posts, with a unique twist: to read a post, users must physically visit the location where it was published.

By blending social media with location-based logic, Proxima aims to bridge the gap between virtual connections and physical presence by encouraging users to discover and share interesting locations. Users are encouraged to discover and interact with new locations, promoting outdoor activity and real-world engagement.

The app provides various features, such as post creation, location-based feeds, commenting, voting, and earning rewards. Users are challenged to visit nearby posts, earning Centauri points as rewards for their explorations. This gamified experience adds a competitive element to content discovery, making it more engaging and rewarding.

Proxima is free for all users, with the primary target audience being social media enthusiasts and explorers.

The goal of the MVP (Minimum Viable Product) is to establish a solid user base, gauging success through metrics like downloads, active users, and user engagement levels. By achieving significant penetration, we aim to attract interest from potential partners and investors for future development.

Proxima is planned to release on Android and IOS.

History

The proof of concept (PoC) built at the end of sprint 10 already contains many of the most important features: user authentication, posts (addition and deletion), comments (addition and deletion), votes, feed, profile cosmetics, challenges, ranking and map.

Doing so, we learnt multiple important things concerning the technologies we used:

- Using **Google Firebase** is a great idea, since it allows to get rid of any trouble concerning the handling of a database, and user authentication.

- **Flutter** is a great framework that really improved our efficiency. Its hot-reload feature notably allowed to quickly test UI changes. Moreover, its testing framework not requiring an emulator made our CI run very efficiently, even with more than 300 tests. The fact that it is cross-platform also makes it an advantage over Kotlin and Jetpack Compose.
- **Penpot** is not a good enough substitute for Figma. We chose it since it is open-source, but it had many issues, such as crashes, slow downs and non fully backward compatible updates. For the MVP, this should be replaced by Figma.

We already have many of the features we want in the PoC for the MVP. The two main features that lack are groups and media support, which will be described in the section "The MVP".

Moreover, substantial changes to the backend are necessary, for security reasons. Instead of relying on client-side logic - which can be compromised on user devices - we should use cloud functions. This approach ensures that only the necessary data is sent to the user's devices, removing the need to trust the client side to correctly filter the data before displaying it.

Analysis of the Situation

Currently, most social media platforms are designed to encourage the sharing and consumption of virtual content.

While this facilitates easy sharing of information, it also confines users to the virtual environment of social media. This can lead to physical isolation from the real world. In extreme cases, users may become dependent on social media, neglecting their surroundings and real-life interactions.

Applications that aim to provide a more outdoor experience through location-based logic are mostly designed for gaming purposes (e.g. Pokemon Go) or specific sports (e.g. running trackers), and do not offer a viable alternative for content sharing. These apps lack the focus on meaningful social interaction and content exchange, limiting their effectiveness in promoting real-world engagement.

Competitors

The current main competitors are mainstream social media platforms (such as Reddit and Instagram) and location-based applications like Geocaching apps. However, these competitors either focus solely on social interaction or on location features in a fully gamified manner. None of them truly combine social interaction with location-based features. Proxima aims to bridge this gap by integrating both elements, offering a unique and engaging user experience.

Complementary Products

The market also offers numerous complementary products to our application. Any product that facilitates travel or encourages outdoor experiences in a passive manner can serve as a complementary asset. For example, travel applications such as Google Maps can assist users in discovering new posts. Similarly, activity trackers can motivate users to engage in outdoor activities and potentially integrate their physical exercise with discovering posts on our app.

The Value Proposition

Proxima provides a solution to the problem of virtual confinement and social isolation by uniquely combining content sharing and outdoor experiences in an intuitive and easy-to-use mobile application. The concept is simple: in Proxima, all content - posts, comments, and interactions - is intrinsically linked to the location where it was created. This encourages real-world exploration and interaction, promoting a healthier balance between virtual engagement and outdoor activities.

On the user side, the incentive for using the application emerges through two main components:

- Sharing content based on a common location-based context
- Discovering content in a treasure-hunter fashion through challenges

Unlike other social networks, where context (meta-content) is often artificially created through threads or hashtags, Proxima ensures that users sharing content have a common "background" based on their physical presence at the same location. This commonality fosters a sense of connection between users.

The second crucial aspect of Proxima is its engaging approach to content consumption. Proxima introduces daily challenges where users must "hunt" for posts to read and earn Centauri points. Additionally, it offers other sources of Centauri points rewards based on interactions such as commenting, variety in posting location, and other forms of engagement.

This gamified aspect creates a friendly competition among users to accumulate the most points. This treasure-hunter experience not only makes content discovery more engaging but also motivates users to actively explore their surroundings.

Regarding customers, Proxima targets businesses where physical interaction is crucial, such as shops and restaurants. Proxima will offer these businesses a location-based advertisement system, bringing several benefits:

- **Relevance**: This ensures that advertisements are relevant to the users who encounter them, as the ads are tied to the users' current location.
- **Proximity**: Because the advertising is location-based, users will be near the business when they see the ad. This significantly reduces the effort

required for users to visit the business, resulting in a higher conversion rate of potential customers for advertisers.

These factors make Proxima particularly appealing to physical businesses, as it allows them to reach an audience that is both relevant and physically able to visit their location, all at a low conversion cost.

In particular, the locality of the advertisement makes Proxima especially attractive for event organizers. By integrating specific time-limited experiences within a certain area, Proxima facilitates efficient promotion of events such as festivals, conventions, and more.

In contrast, traditional social media advertisements are often displayed to users who are hundreds of kilometers away from the relevant location, making the ads less effective. While it is not impossible for other social media to infer position, it is easier in our app and engrained in the user experience. This localized advertisement system contributes to the unique environment that Proxima aims to provide, enhancing both user experience and business engagement/promotion.

The MVP

Personas and Scenarios

Target users:

- **Social Media Users and Content Creators:** Individuals who use social media platforms regularly to connect and share experiences.
- **Travelers:** Those who seek unique local experiences in different cities.
- **Local Explorers:** Residents who want to discover hidden gems and interesting spots in their own city.

Key Persona: Frequent social media users are the primary target due to their high engagement levels and potential to drive content creation and interaction.

High-Level Scenarios:

- **Discovering Local Posts:** Users explore and interact with posts within a 100-meter radius to discover new places.
- **Creating Posts:** Users document and share locations by posting titles, descriptions, and the location.
- **Engaging with Content:** Users engage through comments and votes, increasing interaction.
- **Completing Challenges:** Users earn rewards by visiting nearby locations, enhancing their engagement.
- **Word of mouth**: A user is hanging out with a friend. Looking for any interesting activities nearby, he opens the app to browse posts. Curious, the friend user asks about the app and then tries it out for himself.

User Stories and Key Features

User Stories for Frequent Social Media Users:

1. "As a social media enthusiast, I want to discover and share new experiences through posts around me, so that I can enhance my social outings."
2. "As a social media enthusiast, I want to comment on and vote on posts, so that I can engage with my community and influence which content is most visible."
3. "As a frequent user, I want to be rewarded for my interactions and exploration, so that I can feel my progress."

User Stories for Travelers:

1. "As a traveler, I want to view posts in different cities I visit, so that I can find unique local experiences."
2. "As a traveler, I want to use a map to navigate to posts and challenges, so I can find my way easily in new locations."

User Stories for Local Explorers:

1. "As a local explorer, I want to explore posts from other residents, so that I can find hidden gems in my city."
2. "As a local resident, I want to create local posts, so that people are challenged to explore lesser-known areas of my city."

Key Features:

- **User Authentication**: Necessary for security and personalized user experiences.
- **Post Creation and Location-Based Feed**: The core functionality of sharing and discovering places.
- **Interaction Tools** (Comments and Voting): Enhances user interaction and engagement.
- **Centauri Points and Challenges**: Motivate continued app usage by creating challenges with rewards in form of points.
- **Ranking System**: Encourages competition and engagement by displaying user rankings based on Centauri points.
- **Map**: Allows users to navigate to post locations and to visualize nearby activities.
- **User's posts and comments management**: Allows users to track and manage their activity / virtual exposure on the application.
- **Media Support** (Photos, GIFs, Videos): Enhances post content and profile customization, increasing user engagement.
- **Groups**: Facilitates community building and group exploration.
- **Offline Mode**: Ensures usability even without internet access, improving user retention.

Success Criteria

Evaluation Metrics:

- **User Adoption:** Measured by download numbers and active accounts.
- **Engagement Metrics:** Daily active accounts, daily average interactions per user (creating post or comments, completing challenges).
- **User Feedback:** Direct feedback and satisfaction surveys to gauge user experience.
- **Partnerships and Investor Interest:** Progress in securing support and funding based on app traction and user growth.

Note: key metrics are discussed in more details in the "User Analytics" section.

Benchmarking:

- **User Adoption:** Targeting 3,000 downloads within the first month.
- **Engagement Metrics:** Aim for an average of 1 post per user per day and 3 interactions (votes/comments) per post.
- **User Feedback:** Achieve a minimum 4-star rating on app stores.
- **Partnerships and Investor Interest:** Secure at least five advertisement partnerships with local businesses, as well as meetings with investors within the first three months.

Features Outside the Scope

For the MVP, the focus is on core functionalities that drive user engagement and retention. Additional features will be added in subsequent updates to enhance the user experience and expand the app's capabilities.

1. **Heat Map of Trending Areas**
   - **Reason for Deferment:** Requires a substantial user base to generate meaningful data. Premature introduction could lead to misleading or sparse data displays.
   - **Future Integration:** Aimed for integration once active user thresholds exceed 50,000, providing users with a visual representation of hotspots to explore, thereby driving engagement and exploration.
2. **Comments Tree Structure**
   - **Reason for Deferment:** Reduces complexity in initial development. A linear comment structure simplifies interactions during the early stages of the platform.
   - **Future Integration:** Nested comments will be introduced in a later phase to improve conversation tracking and readability, enhancing discussions and engagement on posts.

Roadmap for features expansion

The roadmap for Proxima includes the following key milestones:

0. **MVP Development**: Focus on core features for initial launch.
1. **Launch**: Release the app in Switzerland, with EPFL students as target users, to establish a user base and gather feedback.
2. **Stability and Performance Improvements**: Address any issues identified post-launch and optimize the app for better performance.
3. **User Growth and Engagement**: Focus on increasing user base and enhancing user interactions.
4. **Feature Expansion**: Introduce additional features like heat maps, and nested comments.
5. **Community Building**: Implement group features to foster community engagement and collaboration.

Non-Functional Requirements

Security, privacy, and data retention policies

Security

User authentication is managed by thrusted third-party authentication systems. Consequently, credential data does not pass through our backend, ensuring that none of this highly sensitive information (such as passwords and tokens) is retained or exposed to any risk on our side. This approach leverages Google's or Apple's robust security infrastructures, providing a secure and reliable authentication process for our users.

Data Retention Policy

Proxima collects only the necessary data required for the proper functioning of the application. This includes the user's representation in the application (such as username, display name, and join date) as well as the content published by the user.

An important consideration is the deletion of accounts as required by law. Our policy in this scenario is to delete all content linked to the user account. Specifically, this includes the user's personal data as well as any content they have published, such as posts and comments. This ensures compliance with legal requirements and respects user privacy.

Location Policy

The primary privacy consideration for Proxima concerns the management of user location data. Recognizing the sensitivity of this information, we have established a strict policy regarding its usage. Proxima will not store any user location data on the backend. The only location data stored in the backend are the locations of posts.

Users know that this value is stored and that other users can see it. This is

typically not an issue since the fact that we were at some place at some point often quickly becomes irrelevant as we move somewhere else. We will, however, make sure that all posts a user has posted are visible on a map, so that if they later decide that the position of a post reveals something about them they wish not to be shared, they can safely delete it. This approach ensures that users' location privacy is maintained while still enabling the core functionalities of the application.

Moderation

Moderation is a crucial aspect of any public social media platform, and our application is no exception. Our policy regarding moderation will be based on a reporting system complemented by AI analysis and human judgment.

If content is reported by multiple users, it will first be evaluated by an AI detection system to determine its appropriateness. If the AI model cannot fully decide on the nature of the content, human moderators will be engaged to make the final decision on whether to remove the content or not. This combined approach ensures that moderation is efficient, accurate, and fair, maintaining a safe and respectful environment for all users. Moreover, since most of the content flagged for human moderation will be ambiguous, this approach benefits the mental health of social moderators by reducing their exposure to shocking or distressing content.

Adoptions, Scalability and Availability

Adoption

1. **Quick Engagement**: Upon the initial launch of the app, users should be able to view and post content with minimal steps to reduce friction and facilitate quick engagement.
2. **Challenge Tour**: When accessing the challenges page for the first time, the user should receive a brief friendly tour explaining how the challenges work and the logic behind Centauri points.
3. **Location Awareness**: Users should be able to view nearby content on a map immediately, highlighting the location-based features of the application.

Scalability

1. **High User Capacity**: The application must scale well to accommodate a high number of users and a large volume of content, ensuring seamless performance as the user base grows. Our infrastructure needs to be ready to support massive adoption spikes that are notorious in new social media platforms adoption curves.
2. **Content Density Performance**: It must also maintain good performance and responsiveness in high content density areas, such as big towns,

monuments, and concerts, to provide a smooth user experience in these environments.
3. **Feature Flexibility**: The backend should be designed to easily support the addition of new features over time, ensuring the application can evolve and adapt to meet changing user needs and technological advancements.
4. **Backward Compatibility**: The backend should be backward compatible to ensure that the addition of new features does not require extensive modifications to the database. The frontend should also be as backward compatible as possible to allow users to take some time before updating the app when a new version is released.

Availability

1. **High Uptime**: Uptime is crucial for any social media platform. Therefore, our application should maintain an uptime above 99.9% to ensure that users have reliable and continuous access to the service.
2. **GPS Signal Handling**: In cases of poor or low precision GPS signal, the application should remain usable. The feed should be refreshed manually rather than on position change, to prevent content from appearing or disappearing unexpectedly, providing a more stable user experience.
3. **Update Pipeline**: The application should have a predefined update pipeline that ensures backward compatibility. This approach will guarantee that all users can access the application seamlessly before and after updates, minimizing disruptions.
4. **Performance and Crash Monitoring**: Anonymous performance and crash metrics should be continuously monitored. This data will help detect and resolve potential issues more quickly, ensuring the application remains stable and performs well for all users.

Functional Requirements

Here are the functional requirements that we believe are necessary for the MVP to be successful.

User Authentication

Users must be able to authenticate into their account to use the app. For this, we will use pre-existing infrastructure, such as Google Sign-in or other common OAuth providers.

Post Creation

Users must be able to create posts. These can contain simple text or media, such as GIFs, photos or videos. These posts will have embedded location data. This will allow the users to share their experiences.

Comment System

Users must be able to leave comments on posts. This will allow them to interact with posts that they see. They should be able to choose between seeing most recent or most popular comments first, as they prefer.

Voting System

Users should be able to upvote or downvote posts and comments. That way, the other users are able to quickly discard posts that received a negative opinion from the community.

Location-Based Feed

Users must be able to see in a feed all of the posts that were made inside of a 100m radius around their current position. This feed will have different sorting options:

- Hottest, which shows the most trending post first using a combination of recency and number of upvotes
- Top, which shows the most upvoted posts first
- Nearest, which shows the closest posts first
- Latest, which shows the most recent posts first

This allows the users to more easily find posts that are interesting to them.

Offline Mode

Users must be able to see as much data as possible even when they are not online. Their user profile, own posts and comments and current challenges should be cached locally and always be accessible.

Posts and comments management

Users should always be able see all of their own posts and comments in their profile page, and delete them if they want to. That way they can have control over their online image and can delete a post if they regret making it. Deleting a post automatically deletes all the comments and any information related to it.

Challenges

Users get 3 challenges every day. These are randomly picked from posts in their area. To complete these challenges, they have to travel to the post and open it. This gives the users a concrete mission that they can do every day.

Centauri Points

Users gain Centauri points by completing challenges. These provide cosmetic improvements on their profile. Users can see how much points they have in

their profile as well as in a leaderboard, where they can compare themselves to everyone. This makes it more interesting to use the app regularly, and improves engagement.

Map Integration

Users should be able to see all the nearby posts, all of their own posts and all their challenges on a map. It should be simple to navigate back and forth from the map to the other pages. It should also be possible to seamlessly open the Google maps app to navigate to any post. This allows the user to precisely see where each post is, and also to know where to go to complete challenges.

Groups

Users that are in the same place should be able create and join temporary groups to use the app together. This will allow them to create posts and complete challenges as a group.

Internal Architecture

The different screen viewmodels will communicate with different repositories to optain the data they need. These repositories talk with the database and obtain the live data. This can be seen in the app architecture diagram, in figure 1 in the appendix.

User Analytics and Acceptance

The primary goal of this section is to understand how users are engaging with Proxima. By tracking specific metrics, we can measure user interaction, satisfaction, and the app's overall effectiveness in promoting real-world exploration through location based content sharing.

Before we fully launch the MVP, it is essential to establish a system to gather pertinent data on user interactions within our application. This infrastructure will enable us to pinpoint areas for enhancement and address any bugs effectively.

Key Metrics

- **Number of Active Users** - Measures daily and monthly active users to assess the app's growth.
- **Number of Posts Created** - Tracks the volume of content generated, reflecting user engagement and the app's usage as a platform for sharing location based experiences.
- **Number of Comments** - Indicates the level of interaction between users, which can help evaluate community engagement.

- Number of **Centauri Points** per User - Provides insight into user involvement in challenges and gamified elements, reflecting the motivational aspects of the app.

Success Criteria

Success will be determined by:

- Effective **user retention** over time. We are aiming to lose less than 10% of our previous active users each month.
- High levels of **user engagement** as indicated by posts, comments, and Centauri points (see Key Metrics above).
- Positive **user feedback** and high satisfaction scores. We are aiming to achieve a user rating above 4 stars on mobile application stores after the first two months (Google Play Store and Apple Store).
- Great overall **posts location coverage**. We want to see a dense concentration of posts with no dark zones (100 meters circle without posts) in large cities within 3 months after the launch of the app in each city.

Analysis Plan

To effectively analyze these metrics, the following strategies will be implemented:

- **Data Collection** - Utilize Google Analytics to track user interactions within the app. This includes taps, posts read, comments made, and post locations visited.
- **Segmentation** - Analyze data based on user demographics (age, post locations, device type) to identify trends and tailor improvements.
- **Performance Benchmarks** - Set specific targets for each metric and compare actual performance against these benchmarks. Include these repetitive performance benchmarks into our regular development pipeline to effectively pinpoint changes that could negatively affect performances.

Note: we also need to make sure that our application stays compliant with data protection rules that apply, like GDPR.

A/B Testing Ideas

- **Post Visibility Proximity Requirements** - Test how varying the radius within which posts can be accessed (e.g., 50 meters vs. 100 meters) affects user engagement and content discovery.
- **Challenges Frequency** - Compare user engagement between different groups exposed to daily versus weekly challenges to determine the optimal frequency for maximizing participation.
- **Notification Styles** - Experiment with different notification styles (e.g., push notifications vs. in-app alerts) when users are near a post location to see which method is more effective in encouraging app interaction.

Design and Implementation

Frontend

Implementation Framework

The application uses the Flutter framework, which is based on the Dart programming language. Flutter can natively compile on Android and iOS from the same code base, which allows us to launch the application to more users easily. We use the Model View ViewModel (MVVM) architecture to ensure structured management and interactions between the UI and the application logic. We use the Riverpod package, which simplifies the application of the Observer pattern, removing unnecessary UI refreshes.

Firebase communication

We use Firebase Authentication to handle user authentication. The Firebase Core, Firebase Auth and third party OAuth packages control the user authentication flow through a third party intent on the login screen.

Storing and fetching the user interactions in the application are handled using the Firebase Core and Cloud Firestore Flutter packages.

User location processing

In order to retrieve the user's surrounding posts and display the map with the current user location, we use the GeoLocator Flutter package. This package simplifies access to the user's current location on every platform. Additionally, we use GeoFlutterFire Plus, which simplifies the storage and fetching of Firebase documents based on their location by leveraging Geohashing.

Essential screens

Login page: First screen that appears when the user is not authenticated. It allows the user to authenticate and access the application.

Feed: The feed is the central screen of the application. Shown when starting the application as an authenticated user, it displays the posts that are at less than one hundred meters away from the user. This screen allows them to sort the feed by hottest, top, latest, or even nearest posts. The feed displays, for every post, its title, the first seven lines of its description, its owner, the number of comments, the date of the post along with its upvote score. It also allows the user to upvote or downvote any posts directly from the feed screen.

Profile page: The profile screen displays all the information associated to the user, such as their username, display name, number of centauri points and badges. Moreover, it displays the title and description of all of their posts along with all of their comments. The user is able to delete any post or comment that they have created, from anywhere, directly from the profile page.

Post creation page: Screen allowing the user to write a new post by setting its title and description. Before posting, the user is able to choose from nearby users with whom they want to post.

Post page: Page displaying all the information associated to a particular post. This screen is displaying the title, complete description, owner, date, distance, upvote score, and all the comments associated with the post. The user is able to sort the comments by top or latest at their convenience, and they can follow up on any comment, creating a thread of comments to discuss and interact about the original post.

Challenge page: The challenge page displays all the currently available challenges that the user can complete alone or as a group. A challenge is described by the post title and distance, along with the time left to complete it. By clicking on a challenge, the user is redirected to the map to localize it more easily.

Group page: The group page allows the user to create or join a group with nearby users. It allows them to post and complete challenges together.

Map page: The map page enables the user to switch between multiple maps that are of two main types. First, we have the maps, allowing the user to see the precise location through pins of their posts and current challenges. Moreover, they are also able to display all the posts that are less than one hundred meters away. By clicking on any pin, the user is able to open Google Map and get a route to it.

Backend

Application Logic

All application logic will be managed on the backend. The client will communicate the necessary information to retrieve the appropriate data it needs to display, and the backend will respond accordingly. For instance, the client will send its position to the backend, which will then provide the nearby posts.

This separation allows the logic to remain flexible and reduces the need for frequent app updates to fix bugs, as these can be addressed on the backend without modifying the client side. Additionally, this approach keeps the frontend lightweight and focused on optimizing the user experience.

Database Interactions

All interactions with the database will be managed through the backend. The frontend will not have direct access to the database; instead, it will obtain the necessary data by communicating with the backend API.

This design choice is primarily for consistency and security reasons. By restricting database access to the backend, we ensure that we can control and predict the actions performed on the database at any given time. This level of control would

not be possible if the client had direct access to the database, as multiple versions of the application could lead to inconsistent interactions with the database. This approach is further justified when considering potential adversarial behavior from the client, where it is crucial to prevent direct database access to maintain security.

Listenable

Some features of Proxima require the client to react to external events. For example, when a user is invited to join a group, the client needs to be notified of this event in real time. This will be accomplished through listenable Firestore documents.

When a listenable logic is required, the cloud function will create a temporary document in a specific collection that the client will listen to. To maintain consistency with our *Database Interaction* policy, the client will only have read access to these documents, and any writes will be performed through backend API calls. This ensures that the client is promptly notified of relevant events while preserving the integrity and security of the backend.

Framework

The backend for Proxima will utilize the Firebase suite. Specifically, the database will be managed through Firestore and the media storage will be held on Firebase Cloud Storage. The backend API will be implemented using Firebase Cloud Functions, written in Node.js as required by the framework. These functions will be callable from the client. This setup provides a scalable and efficient infrastructure for managing the application's backend operations, ensuring reliability and performance.

Data Model

The data for Proxima are stored on Firebase Firestore in a NoSQL, document-oriented database. Additionally, data are cached locally on the client device to allow for offline usage. The data are organized into collections representing the components necessary for the application. Due to the nature of Proxima, the data are tightly tied to the user; therefore, authentication is required to access any of the data. This ensures that user data is secure and accessible only to authorized users.

We now provide the organization of the main data components managed by Proxima.

Users : This collection manages user profile data. The users are stored in a root collection, and for each user, we keep track of the following data:

Centauri points (int), Display name (String), Username (String), Join time (Timestamp), Profile Picture Link (URL).

The distinction between the display name and the username is important. The display name is shown throughout the app and can be the same for multiple users, while the username is unique and used for identifying individual users.

Posts : This collection manages the location-based posts. The posts are stored in a root collection where for each post the following data is stored:

Title (String), Description (String), Publication time (Timestamp), Owner Id (String), Post location (GeoPoint), Post location geohash (String), Number of comments (int), Voting score (int), Media Links (URL List).

The owner id is a string corresponding to the document id of the user who posted the post. The post location geohash is used to perform efficient geo-queries that minimize the number of reads and thus the cost. The media link is a URL referencing the media displayed in the post, stored in Firebase Storage. It can be null if the post is only textual.

Comments : This collection manages the comments under a post. It is a sub-collection nested under each post, and each comment contains the following data:

Content (String), Publication time (Timestamp), Voting score (int), Owner Id (String).

User comments : This collection manages the comments left by a particular user. It is a sub-collection nested under each user, and each user comment has the following data:

Content (String), Parent post Id (String), Publication time (Timestamp).

The document id of a user comment is the same as the document id of the comment under the post. This, combined with the Parent post Id, allows retrieval of the original comment. The redundancy of the fields Content and Publication time is intended to enable fast display of user comments on the profile page, avoiding the need to retrieve each comment's data from under each post. This improves responsiveness and reduces reading costs.

Voters : This collection manages the users that have voted on a post. It is a sub-collection under each post, and each document contains the following:

The user vote type—true for upvote, false for downvote—(bool).

The document id corresponds to the id of the user who cast the vote. This setup allows retrieval of the user's vote state and prevents multiple votes from the same user. The exact same collection structure is also present for comments.

Challenges : This collection manages the active challenges for a user or a group. It is a sub-collection under each user and group, and each challenge is composed of:

Expiring time (Timestamp), Has the challenge been completed (bool), Challenge completion points (int).

The document id of the challenge corresponds to the document id of the post that must be visited to complete the challenge.

Past challenges : This collection manages the past challenges proposed to the user (whether completed or missed). It is a sub-collection under each user, and each past challenge has the same structure as an active challenge, with the fields:

Expiring time (Timestamp), Has the challenge been completed (bool), Challenge completion points (int).

The presence of this collection is justified to ensure that the user is not presented with the same challenge more than once.

Groups : This collection manages the various groups formed by users. It is a root collection, and each group document contains the following data:

Centauri points accumulated by the group (int), Creation time (Timestamp).

Group members : This collection manages the members of a group. It is a sub-collection nested under each group document, and each group member stores the following data:

Joining time (Timestamp).

The document id corresponds to the user id of the member, allowing for retrieval of their data.

Security Considerations

Firestore rules : To enforce the access policy for our database, we will implement Firestore rules. By default, only the backend will have read/write access to the database. As described in the *Listenable* section above, only certain temporary documents will have read access for the client, allowing it to listen and be notified of events. These documents are temporary and will not contain any sensitive data. For all other documents, the client will not have read or write access. This ensures that sensitive data remains secure and that the backend retains control over database operations.

App Check: To ensure that only our application can use our backend API, we will utilize Firebase App Check. This will prevent unauthorized clients from accessing our backend resources. Specifically, it ensures that API calls originate from our authentic application, thereby protecting our backend from misuse.

Infrastructure and Deployment

We will maintain two Firestore databases: one for development and one for production. The code will first be developed and tested on the development database. Once validated, it will be deployed to the production database. This approach ensures that user data is not at risk during development and maintains a consistent production database.

The deployment of the backend API is automatically managed by Firebase, which will allocate resources based on demand. This ensures a scalable backend that can handle varying loads efficiently.

Test Plan

Continuous Integration : The backend will be developed using a continuous integration approach with a minimum code coverage threshold of 90%. This should help us to catch bugs early in the development process and provide confidence to developers when refactoring parts of the codebase.

Cost Tests : For the implementation of new API functions, read/write cost tests will be performed under various conditions to ensure that the functions scale well in different environments. This will allow for better cost estimations and help avoid potential unexpected bills.

Crashlytics : We will utilize Firebase Crashlytics to monitor potential crashes or errors in the application. This will help us quickly diagnose and address potential bugs, ensuring a more stable and reliable user experience.

Timeline/Resource Planning

Execution Roadmap

Planning an efficient schedule will prevent us from making time-wasting mistakes and reduce costs. The timeline is organized over a 14 week period, marked with 4 milestones. Main guidelines will ensure that the whole development process complies with the MVP requirements. A team of 6 people will be in charge of delivering a product ready for initial rollout: 1 full stack developer, 1 backend developer, 2 frontend developers, 1 UI/UX designer and 1 project manager. The major points dealt with during this development process are setup, development, testing and rollout.

Milestone 1 Initial Setup and First Design Iteration

- Setup of the backend (Google Firebase)
- Write description of the backend API
- Basic frontend building blocks
- First interation of UI/UX design (Figma)
- Google login integration
- Tests with mock datas

Sprint/Week Number	Objective	Outcomes
Week 1	Preparation of the codebase	Write description of the backend API. Global view of the project's visual/technical directions.
Week 2	First UI/UX Design and Technical Setup	First UI/UX designs, Google Authentification and Firebase setup, basic frontend building blocks

Milestone 2 Post features development

- Develop the core post system that the users will mainly interact with.
- Ensure that each new feature is thoroughly tested.

Sprint/Week Number	Objective	Outcomes
Week 3	Post creation	Posts with data can be stored and retrieved from the backend. Frontend elements can display this data.
Week 4	Commenting system	Completed commenting feature in frontend and backend
Week 5	Voting system	Developped voting system. Unit testing that ensures that database stays consistent
Week 6	Location-based feed	Combinations of the developped features in the frontend. Mock testing.

Milestone 3 App screens development

- Develop the different pages of the app
- Ensure that each new feature fits well with the rest of the app.

Sprint/Week Number	Objective	Outcomes
Week 7	Profile page	Developed profile page that enables the user to manage his personal data
Week 8	Challenge system	First Gamification of the app.
Week 9	Map page and Open Street Map tile server setup	Possibility to interact with the different posts and feature trough a map.
Week 10	Group page	Users can use the app together.

Milestone 4 Final testing and deployment

- Ensure the stability of the app
- Get first feedbacks from testers
- Release the app to the public

Sprint/Week Number	Objective	Outcomes
Week 11	Quality check	Stable version off the app. Tested and validated user stories
Week 12	Feedback from testers	Pain points, possibilities of amelioration and bug identification.
Week 13	Feedback addressing	Bug fixing, amelioration of the app. Preparation for release on app stores.
Week 14	First public release	Working app ready for marketing and further updates.

Development Resources

Human resources represent the main cost of the project. The backend developer will be in charge of managing the database and the API of the backend. The 2 frontend developers will be in charge of the client part of the app. The UI/UX designer will be in charge of the overall design of the app. The full

stack developper will ensure the integration of all the features goes smoothly. The project manager will be in charge of the coordination of the team and the communication with the users.

Function	Required person-months
Full Stack Developper	4
Frontend Flutter Developers	8
Backend Developer	4
UI/UX Designer	4
Project Manager	4

Deployment Resources

We use the services of Google to manage our backend and to monitor the app. The map will be self-hosted with OpenStreetMap.

Item	Cost / unit	Units	Totals
OpenStreetMap Hosting	200 CHF / month	4	800 CHF
Google Firebase	150 CHF / month	4	600 CHF
TOTAL	1400 CHF		

Maintenance and Upkeep

After the initial rollout, maintaining and updating the app will be crucial to ensure continued functionality and user satisfaction. This phase includes regular monitoring, updates, bug fixes, and improvements based on user feedback. The goal is to keep the app stable, secure, and responsive to user needs. The cost of the paid services and the developers will stay the same during this phase.

Business Model

Revenue Streams

Our business model for Proxima is mainly based on advertising for businesses of any sizes. Our main value proposition is providing localized-based content, allowing businesses of any size to reach their targeted audience. This enables campaigns of any size based on the live location of the user, ranging from worldwide coverage to precise targeting down to districts or even neighborhoods. This can be viewed as a digital version of local billboards.

Our location based content is also able to provide event organizers (such as festival organizers) time-limited event experiences for users within certain predefined areas. This is aimed to be beneficial to the three parties interacting. Our value proposition is to provide participant retention and event discovery. We can

engage the participants through challenges, point-earning and enlarged post reach during the event. This fosters application discovery and user loyalty.

Proxima's business model is based on the density of users interacting in specific areas. We have to scale it according to how our user base grows in each region. We plan to test launch at EPFL (like a mini-city) with local businesses, and expand to new places progressively.

Expected operating Costs

At launch, Proxima's marketing costs will be our primary expenses, as we depend on users' density within the launch area to grow.

Once Proxima starts expanding, as for every social media the infrastructure costs such as hosting, bandwidth, and storage of user interactions and posts make up a significant part of our operating costs. However, our model offers significant advantages, allowing the storage of posts near their physical presence and reducing the cost of it.

We project marketing costs to grow with our expansion to help kick-start user acquisition in new areas. This growth will be accompanied by more operational costs, providing customer and user support, getting in line with local laws, and doing advertiser acquisition.

To keep our application running and increase user engagement, we believe that development, along with research and development, will take a small part of our expenses compared to the two previously cited ones.

We have made the following operating costs estimation, considering an initial launch with 10,000 active users, extensively using all the features available. We have decided that we will host our own OpenStreeMap tile server, to reduce costs as Google Maps is more expensive. We estimate a cost of 200 CHF / month for server hosting. Moreover, we estimate the cost of fetching posts from Firebase, assuming 250 posts read per user per day, at 150 CHF / month.

Item	Cost / unit
OpenStreetMap Hosting	200 CHF / month
Firebase	150 CHF / month
TOTAL	350 CHF / month

Appendix

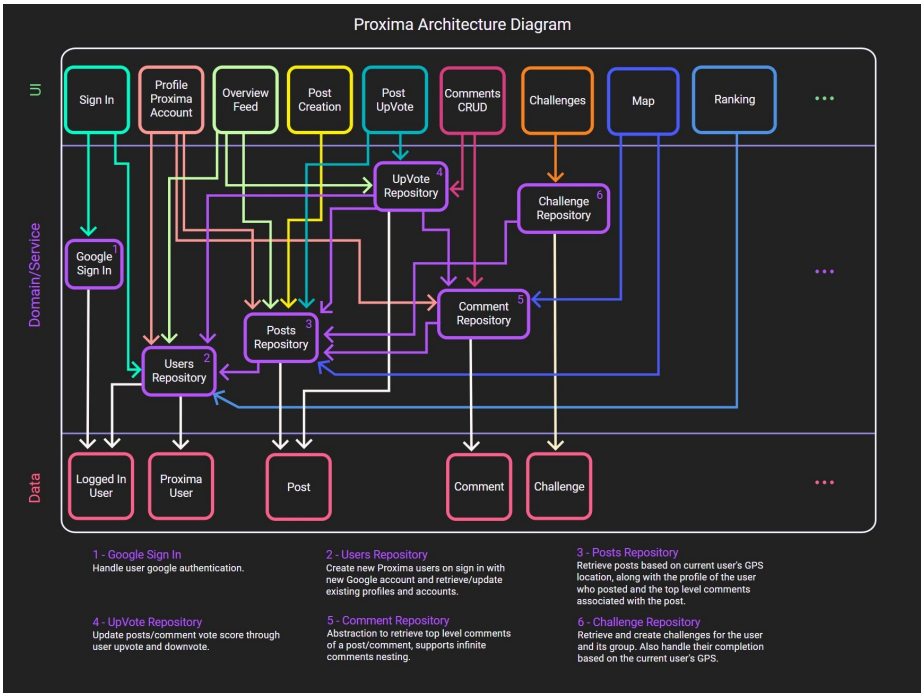

Figure 1: App architecture diagram