Feed App Design Document

DAT250

Eirik Måseidvåg (Student nr: 798179)

Table of content

- **Introduction**
  - Purpose
  - Key Use Cases
- **Domain model**
  - User
  - Poll
  - Invite
  - Vote
- **Use case diagrams**
- **Application flow**
  - Landing screen
  - Poll screen
  - Not found screen
  - Manage poll screen
  - Create poll screen
  - Google Auth screen
- **Wireframes**
  - Landing page
  - Create poll page
  - Manage poll page
  - Poll vote page
  - Not found page
- **System architecture**
- **Conclusion**
- **References**

Introduction

In this initial design and prototyping assignment, we will outline the development of an IoT-cloud software system. The primary aim is to explore IoT and cloud technologies, design principles, and software platforms. This assignment lays the groundwork for a more extensive project in the second part of the course.

Purpose

The primary purpose of this IoT-Cloud Software System is to enable users to create polls and collect feedback via web/mobile applications or specialized IoT voting devices. The system aims to provide an efficient and seamless solution for the collection

and management of responses, enhancing user engagement and real-time feedback.

Key Use Cases

- Poll Creation and Management: Users can create and manage polls through the system. Multiple polls can be defined, and users have the flexibility to handle past, present, and future polls concurrently.
- IoT Device Integration: The system supports the integration of IoT feedback and display devices with specific polls. This feature ensures that each IoT device is associated with the appropriate poll for data submission and presentation.
- Poll Activation: Users can initiate polls, optionally setting time limits. Participants can access polls using unique numbers or provided links, similar to popular platforms like Kahoot. Polls can be configured as either public, allowing open participation, or private, requiring user accounts and login for feedback submission.
- Real-Time Feedback: The IoT-Cloud Software System excels in providing real-time feedback to participants. Vital information, such as current vote counts, is displayed on a web page.
- Poll Closure: Polls can be closed by administrators or users, marking the end of the feedback collection phase. Upon closure, the system processes and stores the results for further analysis.

These key use cases form the core functionality of the IoT-Cloud Software System, offering users an efficient means to create, manage, and participate in polls.

Domain model

The domain model of the IoT-Cloud Software System defines the fundamental entities and their attributes, providing a structured representation of the application's core data. Here, we outline the primary entities and their associated attributes that constitute the foundation of the system. The domain models where created using Figma

User

Attributes:

- id: Unique identifier for each user.
- email: User's email address for communication.
- given name: User's first name.
- family name: User's last name.
- created: Timestamp indicating user account creation.
- updated: Timestamp indicating the last modification of user information.

Poll

Attributes:

- id: Unique identifier for each poll.
- owner: User who initiated the poll.
- title: Title of the poll.
- question: The poll's question or topic.
- status: Current status of the poll (e.g., active, closed).
- isPrivate: Indicator of whether the poll is private or public.
- created: Timestamp indicating poll creation.
- updated: Timestamp indicating the last modification of poll information.

Invite

Attributes:

- poll id: Identifier linking the invite to a specific poll.

- email: Email address of the invitee.
- created: Timestamp indicating when the invite was generated.

Note: Invites rely on the email address, allowing users to invite individuals even if they do not have an account within the system yet.

Vote

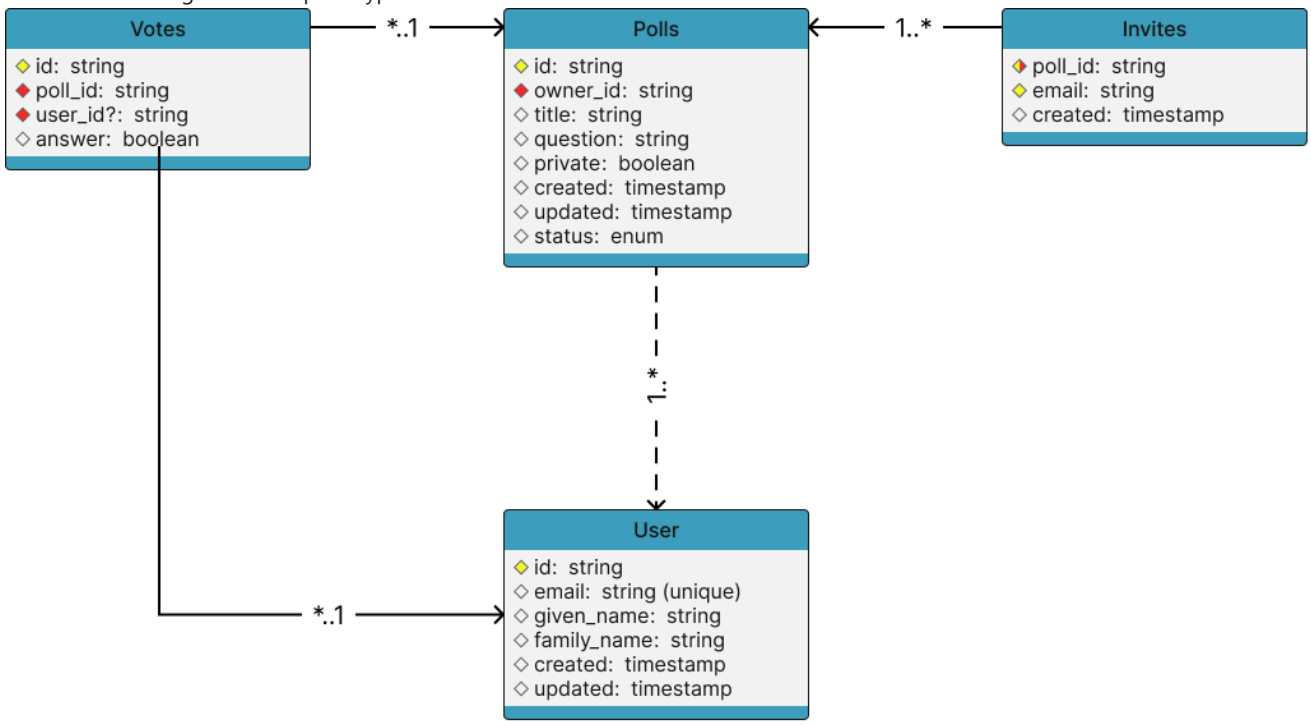
Attributes:

- id: Unique identifier for each vote.
- poll id: Identifier associating the vote with a specific poll.
- user id: Identifier referencing the user who submitted the vote (nullable).
- answer: Boolean value representing the vote (e.g., yes or no).

Note: The user id attribute in the Votes entity is nullable to accommodate voting by individuals who are not authenticated.

This domain model forms the structural basis of the IoT-Cloud Software System, encompassing users, polls, invites, and votes. It captures the essential elements required to manage poll-related data, user information, and participation, providing a clear understanding of the application's data architecture.

Here is an EER diagram of the prototype:



Use case diagrams

For the applications use cases, we will look at two types of users. Anonymous users, and signed in users.

Anonymous users have the following permissions:

- Login with google
- Inspect public polls
- Vote on public polls

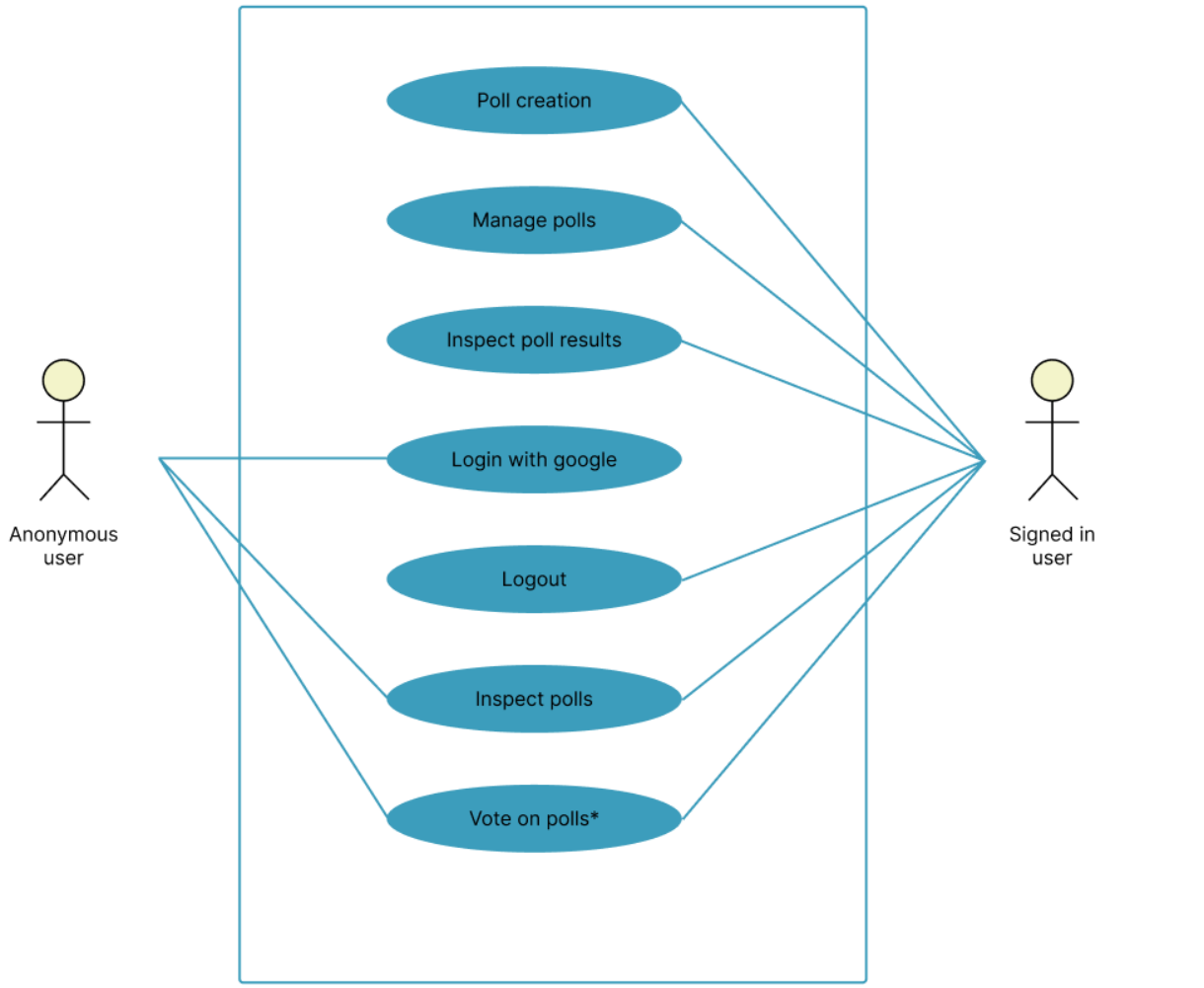
Signed in users will have more options and permissions:

- Logout

- Create new polls
- Manage their existing polls
- Inspect poll analytics
- Inspect public and private polls where they have an invite
- Vote on public and private polls where they have an invite

It should be noted that a signed in users has permission to inspect and vote on their own polls even if it's private and they don't have an invite.

Here is a simple diagram



*: Anonymous users can only vote on public polls, signed in users can vote on public polls + private polls where they have an invite.

Application flow

Our application has the following screens:

- Login screen / Landing page
- Poll screen
- Not found screen
- Manage poll screen
- Create poll screen
- Google Auth

Let's go through each of them and describe more in detail their purposes.

Login screen / Landing page

This will be the screen you will arrive at when visiting the website at the index path. It will allow you to login with google. After a successful login the user will be redirected to the manage polls screen where they will have the option of creating new polls and manage their existing ones. If an authenticated user tries to visit this page, they will be redirected to the manage poll screen.

Poll screen

This is the screen where the voting will happen. If the poll is private, this screen will be behind a guard and require that every visitor has an invite and is authenticated through Google. If a user tries to access this page without being logged in, they will be redirected to Google's authentication page. If the login is successful, they will be redirected back to the poll they initially tried to access.

If a user who is logged in tries to access a poll that is private, there will be a guard to ensure the user has an invite to the poll they are visiting. If they do not have an invite, they will be shown the "Not Found" screen. If a user tries to visit a poll that doesnt exist they will also be shown the "Not Found" screen, i.e. the user will purposefully not know if they lack an invite to the poll or if it doesn't exist.

Not found screen

A simple screen with a "Poll not found" text.

Manage poll screen

This screen will be behind an authorization guard, requiring the user to be logged in to inspect it. Here the user will be able to inspect and manage their polls and create new ones.

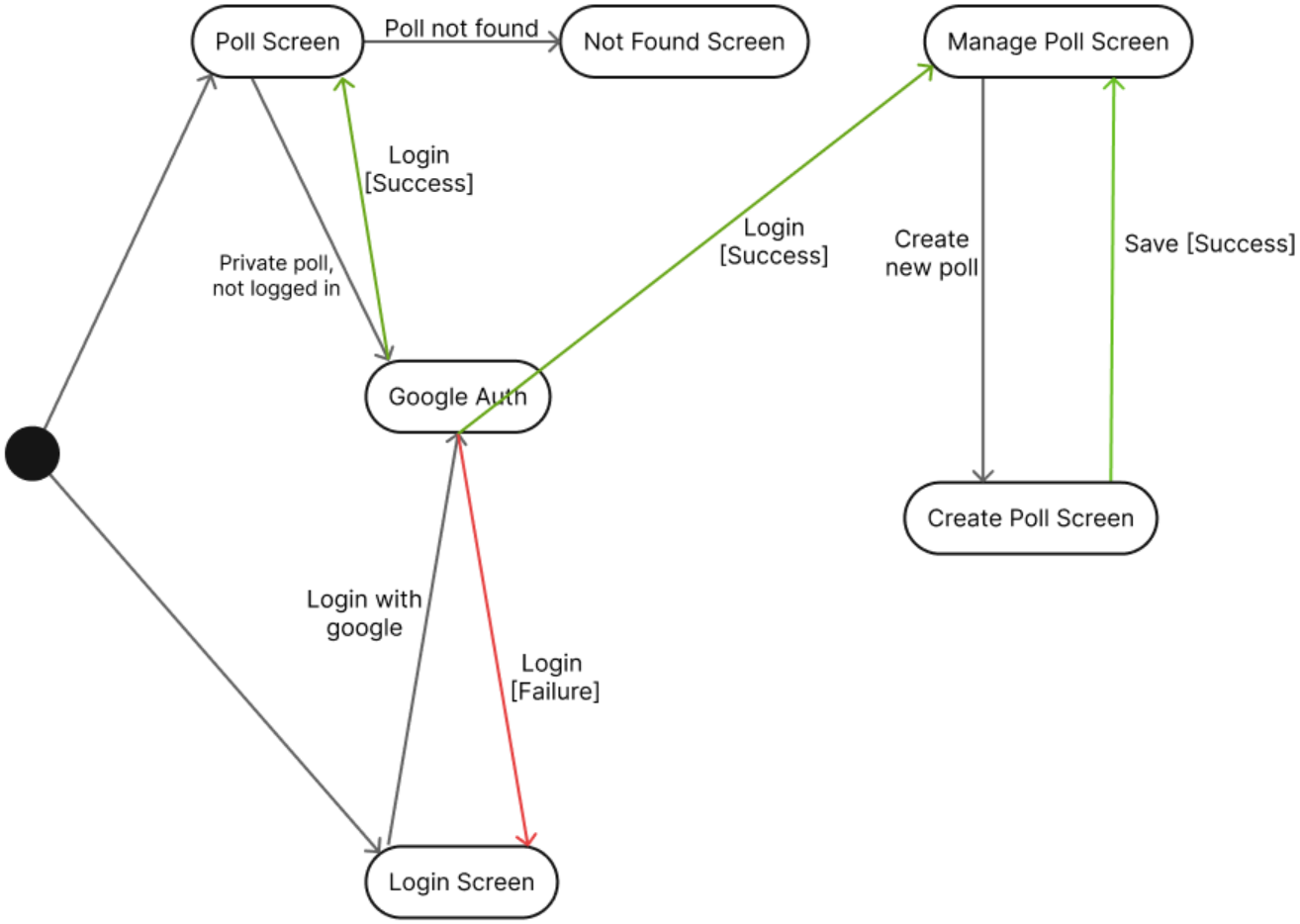
Create poll screen

This screen will be behind an authorization guard, requiring the user to be logged in to inspect it. Here the user will see a simple form to create a new poll. They will here be able to invite people by email if they choose to make their poll private.

Google Auth screen

This is Google's own screen. It provides the user the option to sign in with google. It is not a part of FeedApp, but used as a third party.

Here is a simple diagram of the application flow described:



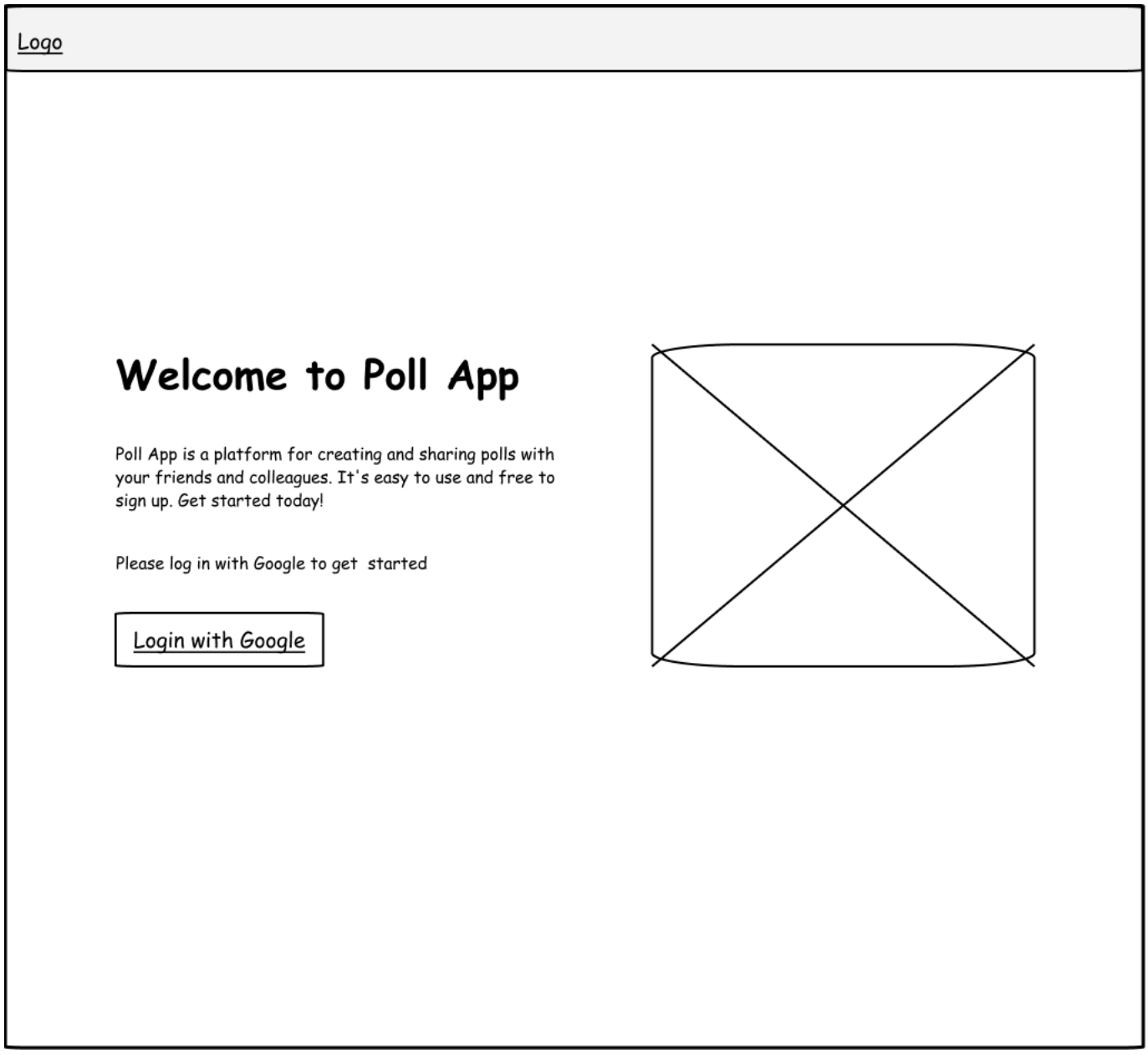
Frontend wireframes

Each page will have navbar with the logo on the left side. If the user is logged in, they will be able to see their name and google profile picture on the right. There will also be the option to log out in a menu dropdown. If the user is not authenticated, they will be shown a login button instead. On the landing page there will not be a login button in the navbar as there will already be one in the body of the page.

These wireframes were created in Figma with the help of a plugin called WireGen.

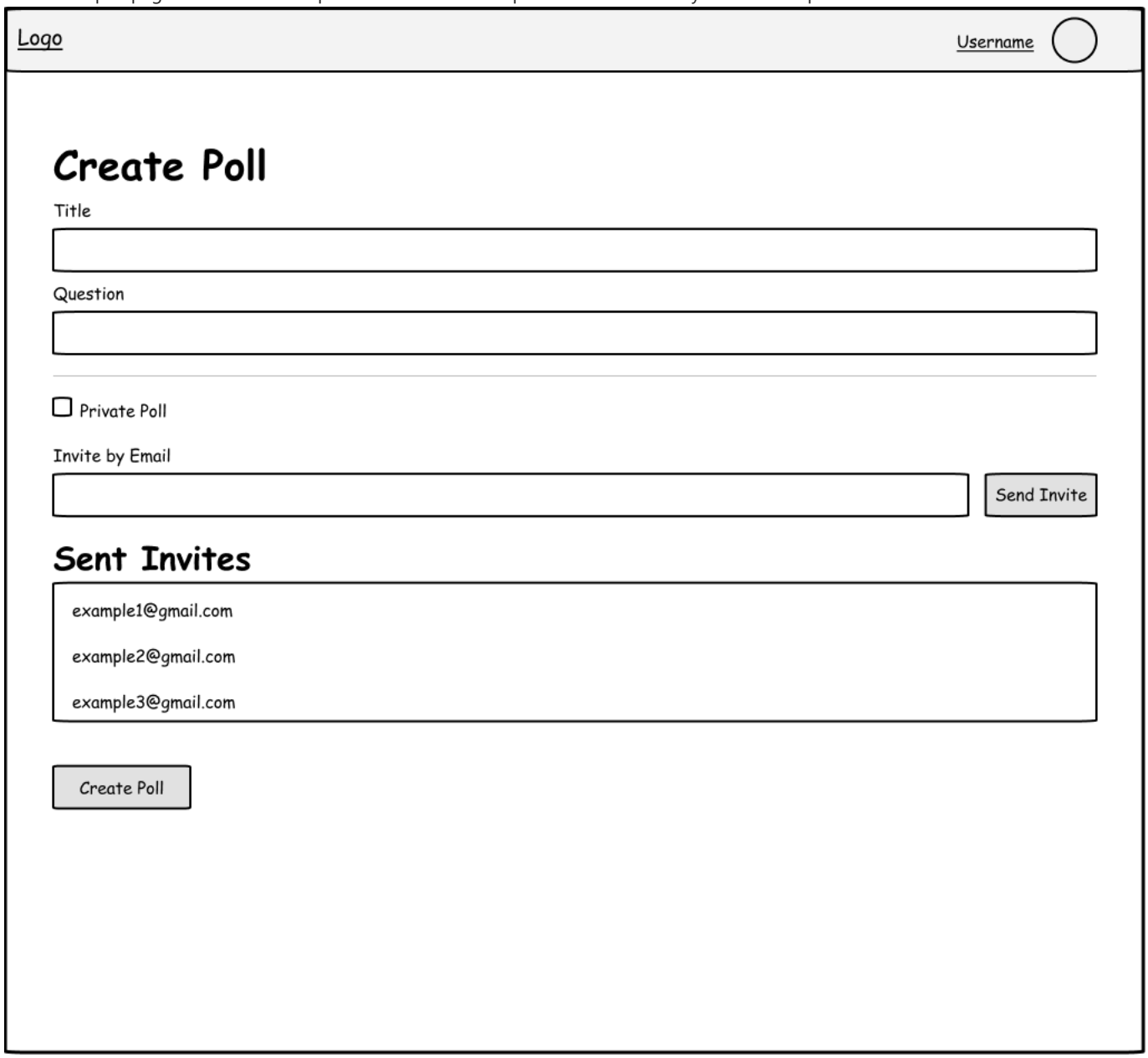
Landing page

The landing page will be simplistic. It will show a brief welcome with some description of what the app is about as well as a login button and an illustration.



Create poll page

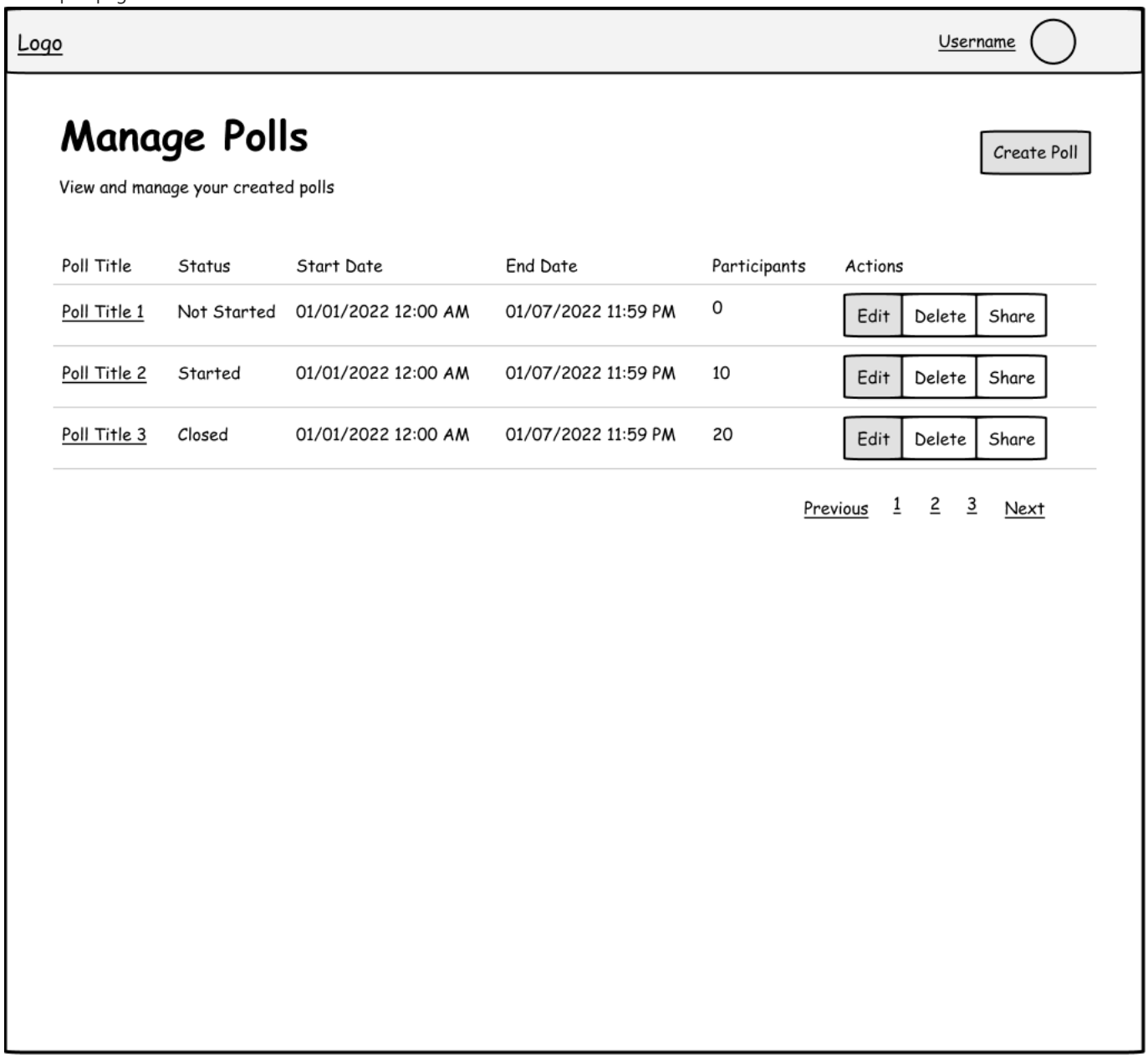
The create poll page will include a simple form to let the user provide the neccessary data for the poll.



Manage poll page

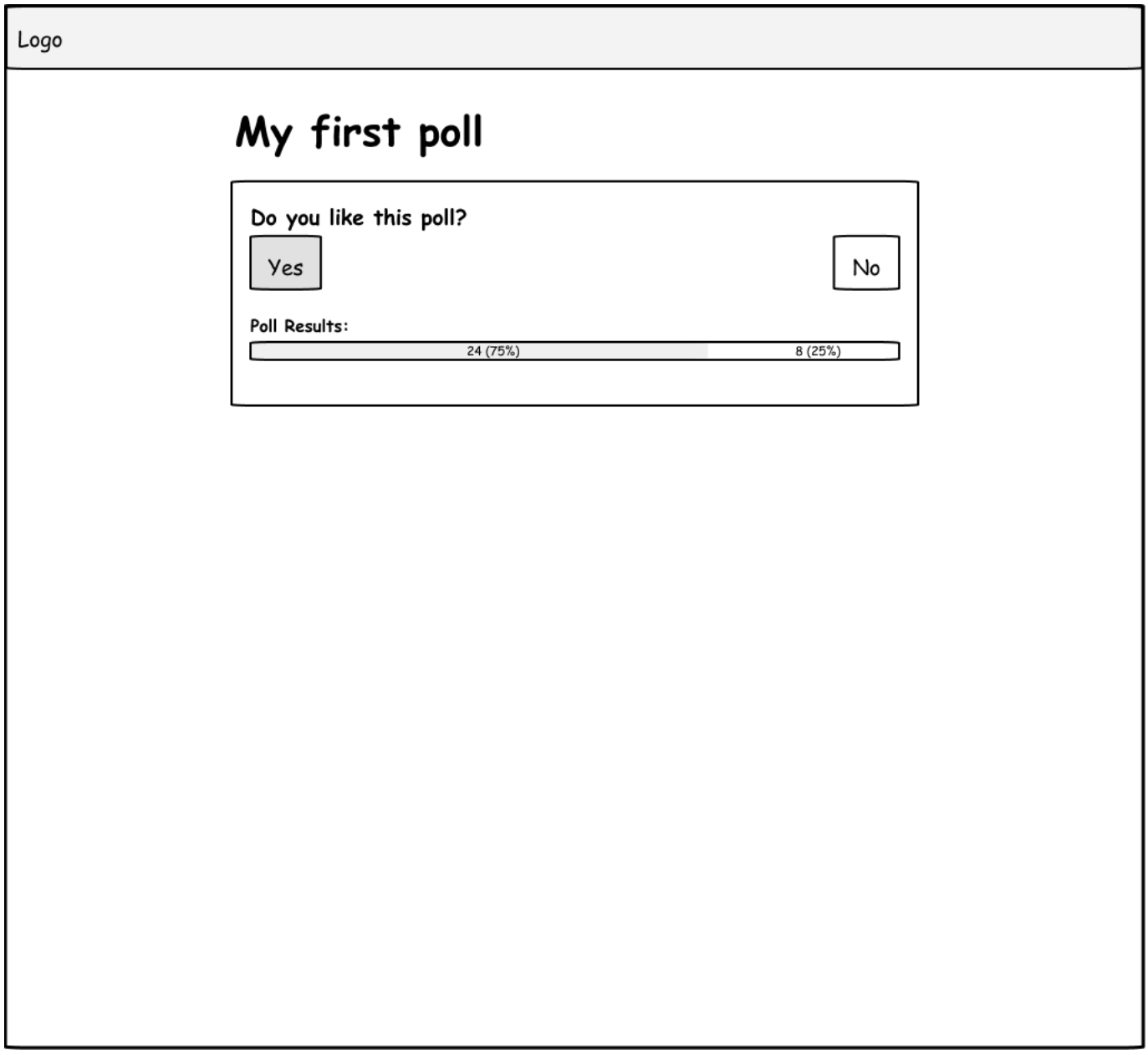
The manage poll page will let the user inspect, edit, delete and share their polls. Poll analytics will be shown for each of them, and a pagination bar will be displayed if the amount of poll exceeds an undecided number. There will also be a button to take the user to the

create poll page.



Poll vote page

This page will show the user the poll title, question, and options to click yes or no. There will also be a simple illustration underneath to show the percentages of the current vote results. If the poll is closed, the yes/no buttons will be disabled.



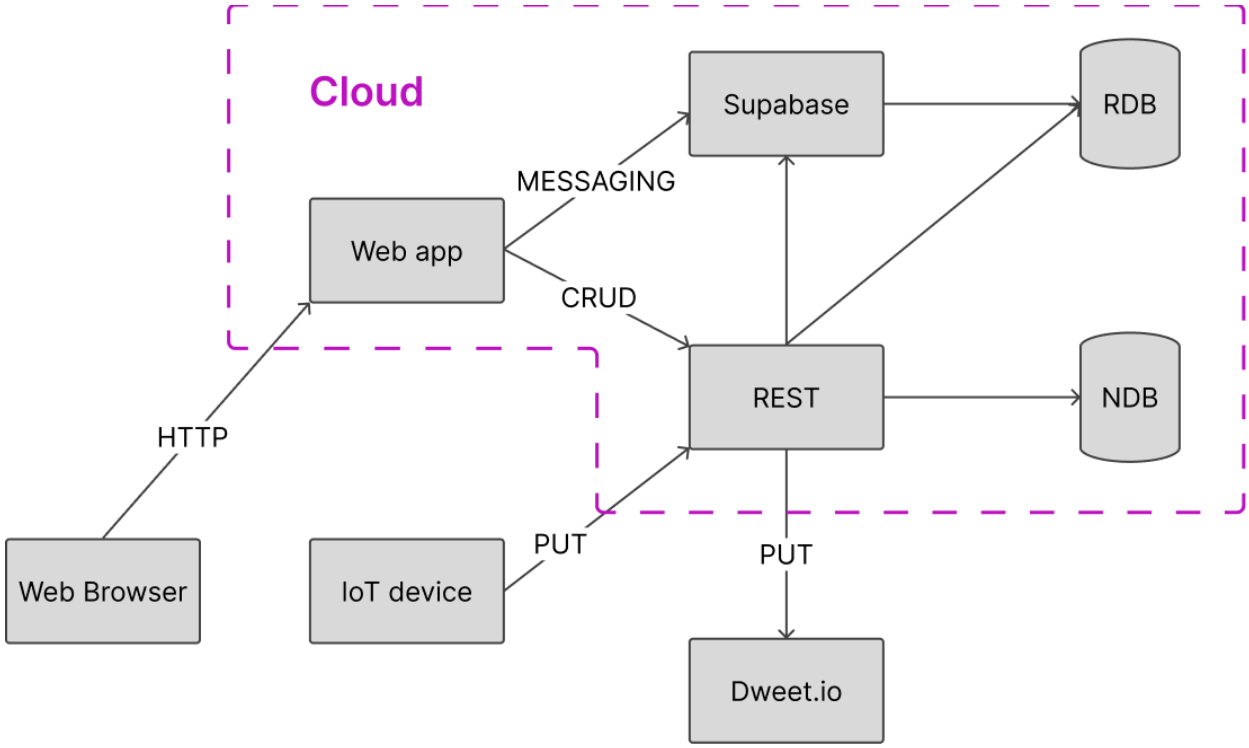
Not found page

This is a simple page that let's the user know that the resource they were looking for was not found.

System architecture

Here are the following containers that will be used to create this system:

- Web application: TypeScript frontend app that will be served in the cloud. We will be using Vue.js version 3 with libraries such as tailwind for styling, eslint and prettier for linting and supabase for real-time communication with the poll votes.
- REST API: Core application to handle the buisness logic. Will be written with Nest.js using TypeORM to communicate with the relational database. Updating votes will happen through supabase to allow for real-time subscription in the web app.
- Supabase: Third party SAAS that will handle real-time communication and authentication. Will only have access to the user and votes tables.
- RDB: Relational database. Will use PostreSQL. Stores all information about users, polls, invites and votes. Served in the cloud.
- NDB. NoSQL database. Will use MongoDB. Stores aggregated analytics data of closed polls.
- Dweet.io: Messaging service where aggregated analytics data will be posted.
- Web browser: Local browser to be used to open the web application. Can be any browser - desktop or mobile.
- IoT device: Simulated device that will communicate with the core buisness logic and allow for simple voting.



Conclusion

In closing, the design document for the IoT-Cloud Software System has provided a comprehensive blueprint for the development and implementation of this innovative platform. Throughout this document, we have meticulously explored the system's purpose, key functionalities, domain model, and architectural considerations.

As we move forward, the insights gained from this design document will guide the subsequent stages of the project. The collaborative efforts of the development team, guided by this document, will shape the realization of this system in the second part of the course. We acknowledge that the system's successful implementation will require careful planning, iterative design, and rigorous testing. The development methodology, inspired by IoT and software architecture principles, will serve as a valuable guide.

In conclusion, the IoT-Cloud Software System is poised to offer a dynamic and engaging user experience, bringing together IoT and cloud technologies to streamline the process of gathering valuable user feedback. We look forward to bringing this vision to life and realizing the full potential of this exciting project.

References

1. [Vue.js (https://vuejs.org/)](https://vuejs.org/)
2. [Figma (https://www.figma.com/)](https://www.figma.com/)
3. [supabase (https://supabase.com/)](https://supabase.com/)
4. [Nest.js (https://nestjs.com/)](https://nestjs.com/)
5. [PostgreSQL (https://www.postgresql.org/)](https://www.postgresql.org/)
6. [MongoDB (https://www.mongodb.com/)](https://www.mongodb.com/)
7. [Google OAuth2 (https://developers.google.com/identity/protocols/oauth2)]
   (https://developers.google.com/identity/protocols/oauth2)