CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

FrontDP
A Privacy-Aware Front-end for WebDP

Bachelor's thesis in Computer science and engineering

Andres Arriagada Fuentes
Ann Heijkenskjöld
Kristin Hulling
Eric Källman
Wilma Nordlund

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Bachelor of Science Thesis 2024

FrontDP
A Privacy-Aware Front-end for WebDP

Andres Arriagada Fuentes
Ann Heijkenskjöld
Kristin Hulling
Eric Källman
Wilma Nordlund

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

FrontDP: A Privacy-Aware Front-end for WebDP

Supervisor: Alejandro Russo, Department of Computer Science and Engineering
Examiners: Arne Linde and Patrik Jansson, Department of Computer Science and Engineering
Graded by teacher: Niklas Broberg, Department of Computer Science and Engineering

Bachelor of Science Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

FrontDP: A Privacy-Aware Front-end for WebDP
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

The technological advancements of today have made personal data a valuable commodity. There are several purposes for utilising personal data, including tracking and profiling of specific individuals. A mathematical technique known as differential privacy makes it possible to share data about a group of individuals while maintaining the privacy of their personally identifiable information.

DPella is a Gothenburg-based company that has built a RESTful API called WebDP. The goal of their product is to connect different open-source differential privacy engines. This Bachelor of Science project aimed to build a front-end application for WebDP called FrontDP. There are two main users of the application, namely the data curator, and the data analyst. This application would allow data curators to share datasets and assign analysts. Additionally, data analysts could perform analytical queries to derive information, while keeping the identifying information about individuals in the dataset private.

The group explored building on a separate open-source front-end application before deciding to build their own using React. Prior to the front-end development, a design phase was conducted with the layout being based on UI patterns. Before implementing the front-end, a prototype was developed and tested to ensure that the design was comprehensible and easy to use. The end result was a front-end application that covers most of WebDP's API. However, further development is required to ensure the application's longevity.

Keywords: front-end development, web application, differential privacy, API, user interface, accessibility, user friendliness, design patterns.

Sammandrag

Utvecklingen inom den tekniska världen har gjort persondata till ett värdefullt verktyg. Det finns många syften med att använda persondata, exempelvis spårning samt profilering av specifika individer. Den matematiska tekniken differentiell integritet möjliggör delning av data gällande en grupp individer utan att deras personliga information riskerar att bli utsatt.

DPella är ett Göteborgsbaserat företag som har byggt ett RESTful API som heter WebDP. Målet med deras produkt är att koppla samman olika motorer som använder sig av differentiell integritetsteknik och som har öppen källkod. Detta kandidatarbete strävade efter att bygga en front-end-applikation till WebDP som kallas för FrontDP. Det skulle finnas två huvudsakliga användare av applikationen: datakuratorn samt dataanalytikern. Den här applikationen skulle tillåta datakuratorer att dela dataset och tilldela dem till analytiker. Dessa skulle i sin tur kunna genomföra statistiska förfrågningar på dataseten för att utvinna information, samtidigt som man bevarar integriteten för den personidentifierande informationen.

Gruppen utforskade möjligheten att bygga vidare på en redan existerande front-end-applikation med öppen källkod innan det bestämdes att bygga en egen produkt med React. Inför front-end-utvecklingen genomfördes en designfas med hjälp av UI-mönster. Innan implementationen utvecklades en prototyp som testades för att säkerställa att designen var begriplig samt lättanvänd. Slutresultatet var en front-end-applikation som täcker det mesta av WebDPs API. Däremot krävs ytterligare utveckling för att säkerställa applikationens hållbarhet.

Acknowledgements

We would like to thank Alejandro Russo, who not only gave us help and feedback throughout the project as our supervisor, but also acted as a bridge between us and the team at DPella. Furthermore, we would like to thank the team members at DPella who have answered our questions and shared their knowledge with us. We would also like to thank the two developers of OpenDP at Harvard University, Ellen Kraffmiller and Chuck McCallum, who gave us invaluable input at a pivotal point in the process of the project.

Finally, we would like to thank the individuals at the Department of Communication and Learning in Science at Chalmers, who gave important feedback on the report both in the early and later stages of the writing process.

Andres Arriagada Fuentes, Ann Heijkenskjöld
Kristin Hulling, Eric Källman, Wilma Nordlund

Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

API	Application Programming Interface
DP	Differential Privacy
REST	Representational State Transfer
MVP	Minimum Viable Product
WCAG	Web Content Accessibility Guidelines

Contents

1
Introduction

The demand for cybersecurity solutions increases as digital advancements continue. Nowadays, not only regular IT systems collect data about their users. With the broadened use of vast machine learning algorithms, every individual's personal data might be in danger of being gathered and stored during the learning process, and so restrictions in data sharing are introduced [1]. Our personal data is a valuable resource for multiple stakeholders with different motives [2]. It could be used for a plethora of objectives, such as choosing appropriate advertisements for the individual in order to increase sales, or forecasting future trends. Whether it is a company with economic interests or a researcher with academic intentions, the risk of tracking which individual the sensitive information relates to should be eliminated.

1.1 Project Background

One way of protecting the privacy of individuals and concurrently making their data useful is by using Differential Privacy. It is a technology that uses statistical noise, a mathematical phenomenon, to protect data privacy. The technology disturbs the result and data insights in order to protect individuals' privacy. Consequently, if the demand for privacy is high, inaccuracy of the presented data analytics will increase due to injection of noise. However, the noise has a smaller impact on accuracy as the dataset's size increases [3].

1.1.1 Differential Privacy and DPella

Differential Privacy (DP) is the core foundation of DPella, a Gothenburg-based company founded in 2020 that proposed this bachelor project [4]. Additionally, they have initiated an open-source project, WebDP, which is an interface for the usage of DP engines. Their primary goal is that their interface will enable other developers and systems to interact seamlessly with various DP systems over the web in a transparent manner.

There are two main types of user profiles that will utilise the interface: data curators and data analysts. Data curators are responsible for uploading the datasets, setting privacy parameters, and assigning data analysts to the datasets. Meanwhile, data analysts are tasked with analysing the privacy-protected datasets. In addition, there is also an admin role who is responsible for user management.

Furthermore, there are several full-stack open-source products that have been launched. Two of these products are OpenDP [5] and Tumult Analytics [6], both of which utilise the theory of DP. However, in contrast to them, DPella's vision is to use all open-source DP services, not only their own. The primary reason for the gathering is to eliminate the need for the data analyst to understand which engine is best suited for them. Instead, WebDP will recommend an engine depending on the analyst's demands and the attributes of each machine. In addition, developing a product that collects and utilises all engines might also standardise the future development of DP services. As a result, a standardised easy-to-use application where one can choose among multiple DPs is desired.

WebDP is implemented as a Representational State Transfer (REST) Application Programming Interface (API). The purpose of an API is to declare the way of communicating between different systems. It enables this by setting strict rules and protocols for how the format of both the requests and its data should be. APIs are widely used in web development in order to facilitate for developers to set up communications with a system that they are not familiar with. Instead of learning the other system's internal workings in depth, the developer can utilise the system's API in order to use its services. A RESTful API is a type of API that complies with the principles of REST, which is an architectural style for designing networked applications. These APIs use standard HTTP methods and typically use JSON or XML as the data format for information exchange [7].

1.1.2 The Need for a User Interface

A crucial part for any company is to be able to present and provide their services, and what DPella's open source project currently lacks is a solution for this. The bachelor's project team has been asked to provide DPella with a user interface that enables the startup to continue developing and providing their WebDP as a solution. The front-end application should be comprehensible, secure, intuitive and user-friendly, so that individuals involved with the subject can navigate it effortlessly and feel safe while exploring.

A user-friendly front-end relies on the implementation of suitable design patterns. These design patterns should be familiar to users and enable easy navigation, comprehension, and should never surprise the user [8]. By incorporating established design patterns that align with client and business needs, it is possible to develop a front-end that not only enhances the user experience but also instills a sense of security and confidence in users.

To create a user-friendly front-end, it is essential to adhere to guidelines that guarantee inclusion and equal access for all users. Developers can promote equality and prevent exclusion by using these guidelines to create digital products that are usable by individuals with disabilities. Following accessibility guidelines signifies an endorsement of ethical values, including fairness, justice, and respect for human rights. It is the responsibility of front-end developers to prioritise accessibility and

consider the diverse needs of users, ensuring that everyone can access and interact with digital content without barriers or discrimination. One way of doing this is by adhering to the global Web Content Accessibility Guidelines (WCAG) [9].

1.2 Purpose

The purpose of this bachelor's project is to develop a user-friendly front-end for WebDP that follows design principles and is comprehensible to the user.

1.3 Goal

The goal of this project is to build the front-end so that it implements as many building blocks as possible from the API regarding handling and analysis of data, for the two main kinds of users of the interface, namely the data analyst and the data curator. Additionally, the front-end should be user-friendly, and the data insights service requires careful consideration of design patterns and simplicity in navigation. By implementing these principles, it is possible to develop a front-end that not only facilitates ease of use but also instills confidence in users.

1.4 Boundaries

The project will maintain a clear boundary at the API level. The team's responsibilities will be limited to the client-side development, ensuring that the front-end is intuitive, thus allowing users to interact with the DP functionality provided by the API. The project does not include server-side development, modification of the API, or any work related to back-end and database management. Furthermore, this report will not include detailed explanations of the technologies behind DP. By setting these boundaries, the effort will be targeted towards delivering a high-quality front-end, without delving into the back-end complexities.

2
Theory

Front-end development is the practice of building the user interface of a website or application [10]. With the use of programming languages such as HTML, CSS, and TypeScript, the developers of the front-end build the visual and interactive elements the users interact with on their devices. The front-end developers are responsible for ensuring that the website or application has a pleasant appearance, while at the same time being easy to use, and functioning correctly. Furthermore, the developers work closely with the designers and back-end developers in order to integrate the front-end with the back-end systems that power the website or application, whilst maintaining the design of the interface.

In order to deliver a functioning front-end, developers have a general process to follow. However, this can vary depending on the project requirements and the preferences of the developers, but some of the steps are present in most processes [11]. These steps will be presented in this chapter.

2.1 Defining the Requirements

One of the first steps in front-end development processes is to define requirements, scope, and users, as this lays the foundation for building a successful website or application.

2.1.1 Requirements

Defining requirements involves identifying and collecting information about what the website or application needs to accomplish [12]. This can involve functionalities, features, user interactions, and any type of technical specification. The goal is to maintain a flexible and evolving list of demands that guide development, and to further guarantee that the project satisfies user needs and expectations. Once the developers have gathered the information, it is analysed and further specified, in order to ensure that the requirements are clear and consistent. The requirements are then documented in a structured format that serves as a reference for the development team throughout the process.

2.1.2 Scope

Defining the scope involves identifying the boundaries of the project, so that from early on it is clear what should be included and excluded from the project [13].

This helps in setting expectations regarding the deliverables, which in turn provides every party that has some interest or involvement with the product a collective understanding of the project.

2.1.3 Users

Understanding the preferences of the user is crucial when designing and developing a user-friendly interface. When analysing the website or application from the end users' point of view, the developer team can identify user requirements such as certain functionality regarding for example accessibility, responsiveness, and performance. By having defined requirements, scope, and users, the development team can ensure that the product meets the needs of its users and delivers the expected outcome. The creation of personas is one method for understanding the needs and preferences of the users [14]. User personas are fictional representations of the different types of users who will interact with the website or application. The fictional representation can include information such as demographics, behavioral patterns, goals and needs of the user, and much more.

2.2 Programming Languages and Frameworks

One important decision is selecting an appropriate programming language. The choice of programming language can affect everything from the development timeline to the product's future scalability, maintainability, and performance [15]. Many aspects need to be taken into consideration, such as the experience and expertise of the team, the language's compatibility with other technologies, and industry standards, to name a few. Another critical aspect is determining the framework. There are frameworks such as Angular and Vue that provide pre-built components and libraries, which can streamline the development process, as well as the user experience of the final product.

2.3 Designing and Implementing the Interface

Based on user personas and requirements, the designers can create prototypes of the interface and experience [8]. Before creating more detailed prototypes, the first step is usually to draft a basic sketch to create an overview of the design [16]. The prototypes will be evaluated, improved, and later on they can serve as blueprints for the development team to implement [17]. In this part of the planning process, there are also choices made regarding aspects of the design such as responsiveness and accessibility. Although it is common to collaborate with designers to create mockups and prototypes that help visualise the final product, this is not the case in every development process. If the development team has knowledge about the design of a user interface, this part of the process can be conducted within the development team.

2.3.1 User-Friendly Design

The utilisation of appropriate design patterns plays a vital role in creating a successful front-end application [18]. These patterns, which can vary depending on the specific requirements of clients and businesses, provide a solid foundation for the development process. The following images display a variety of design patterns, in order to clarify what a design pattern is.

Feature, Search, and Browse. Feature, Search, and Browse are patterns commonly used in eCommerce websites. This pattern features a search bar, featured items, and menu items to the left for browsing. See Figure 2.1.

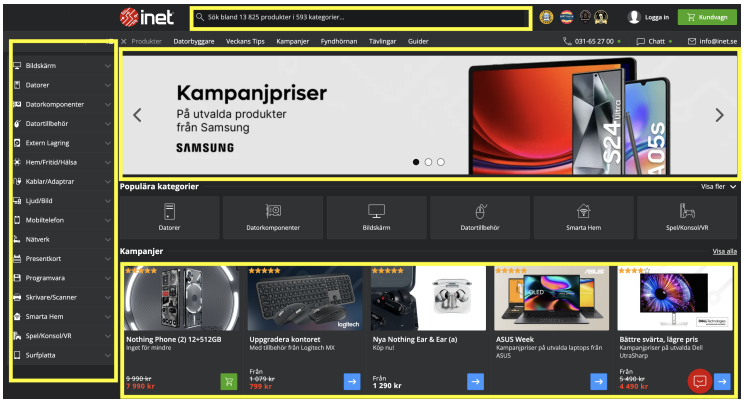


Figure 2.1: Screenshot of the website *Inet*.

Dashboard. The dashboard pattern is commonly used in community based platforms, and serves as a homepage with content that gets updated frequently. See Figure 2.2.

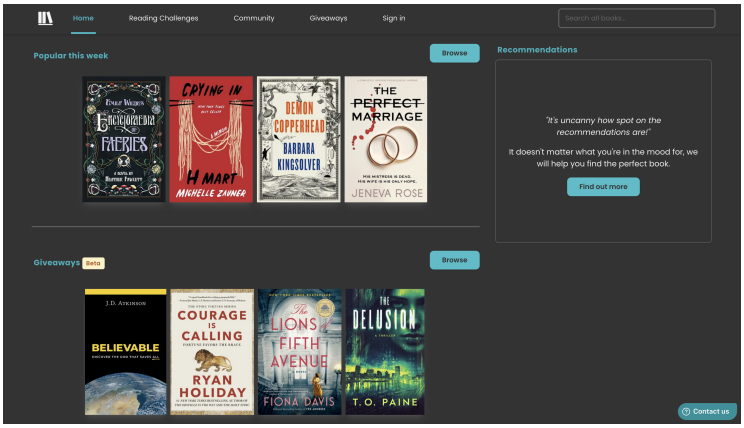


Figure 2.2: Screenshot of the website *The Storygraph*.

Wizard. A wizard can be utilised when the user wants to achieve a single goal which can be broken down into dependable sub-tasks. The wizard pattern is often

used during transaction processes, as well as information upload processes. See Figure 2.3.

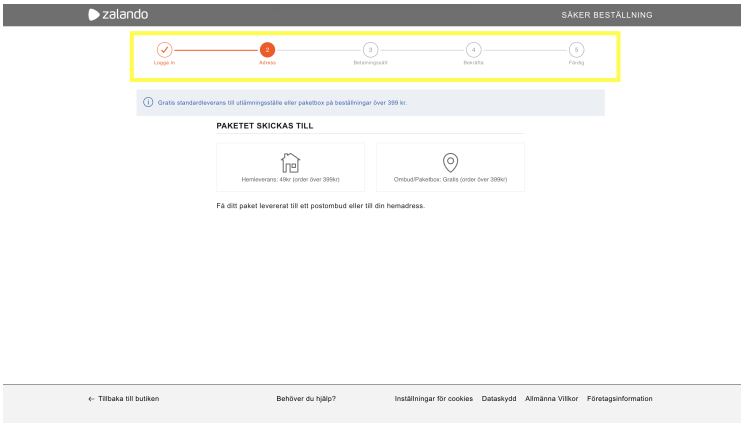


Figure 2.3: Screenshot of the website *Zalando*.

Escape Hatch. Escape Hatch is a pattern that is common across a multitude of different websites. It is implemented on every page and is recognisable over the whole web application. Pressing the escape hatch results in the user usually returning to the homepage or start page. The convention is that the company or site logo on the top left serves as an escape hatch. See Figure 2.4.

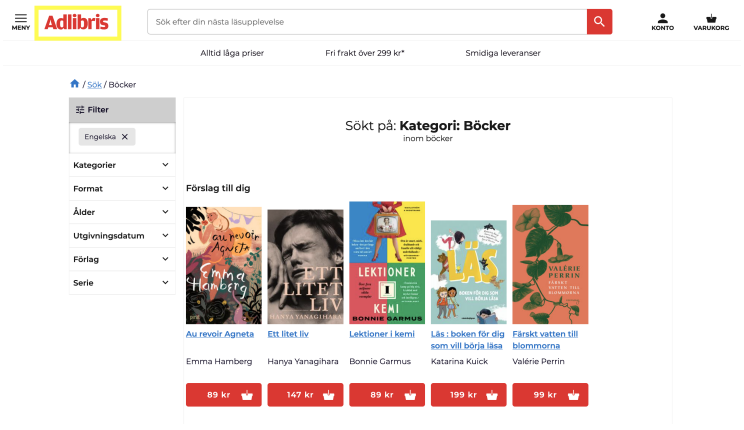


Figure 2.4: Screenshot of the website *Adlibris*.

Sign-In Tools. Websites that provide user functionality often display this with an icon at the top of the page, signaling that there are tools for signing in, signing out, and managing your profile. See Figure 2.5.

Visual Framework. The visual framework, which is also known as the graphic expression, is the design framework of the website. Reusing colors, fonts, icons, and patterns over the different pages creates consistency and recognition. See Figure 2.6 and Figure 2.7 for how a visual framework can be represented over different pages.

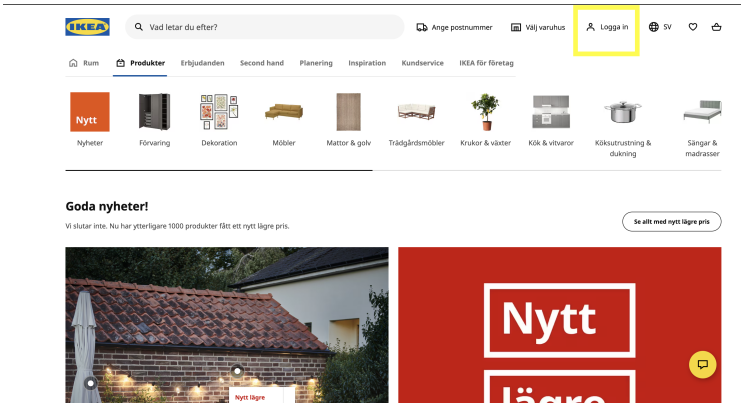


Figure 2.5: Screenshot of the website *IKEA*.

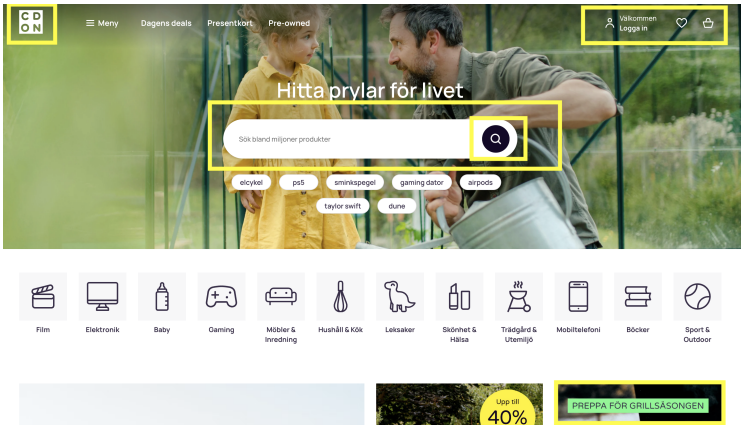


Figure 2.6: Screenshot of the homepage on the website *CDON*.

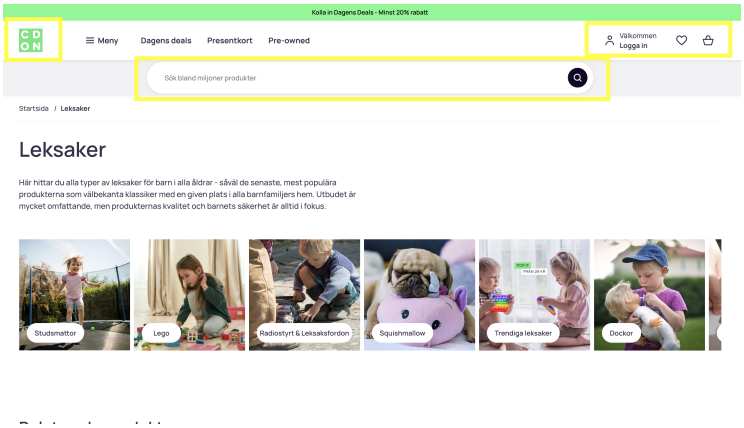


Figure 2.7: Screenshot of a category page on the website *CDON*.

Prominent Done Button. By making an action button look visually different than other buttons, the user becomes aware that something will happen when the button is clicked. This pattern is typically used for sign-in buttons, buttons that create accounts, buttons that initialise downloads, and buttons that confirm transactions. See Figure 2.8.

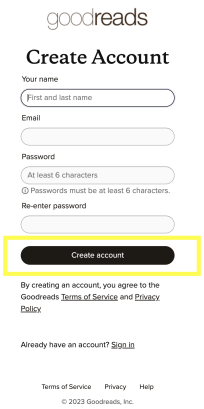


Figure 2.8: Screenshot of the website *Goodreads*.

2.3.2 Architecture and Infrastructure

Defining the project's architecture and infrastructure is another vital part of the planning process [15]. This can also include a variety of aspects, some of which are important for most front-end projects. These include the hosting environment, build tools, and strategies for deployment. In this step, it is important to factor in future scalability and maintenance requirements. Other aspects that need to be decided upon are coding standards and naming conventions, as well as project management methodologies.

2.3.3 Implementation

Implementation of the interface is conducted when the design is complete. This step involves programming in a chosen language to realise the components of one's design, resulting in a closer version to the final product. User Stories can be converted into tasks for which developers can take the responsibility of completing.

During the implementation process of a front-end, various types of tests can be helpful in reducing bugs in the system. The testing structure is commonly described as a pyramid, with unit tests at the bottom and UI tests at the top. Unit tests are small tests focused on individual components, while UI tests should evaluate the user interaction flows. The number of unit tests should exceed the number of UI tests, hence the pyramid shape [19].

2.4 Usability Testing

Incremental feedback and continuous version updates are vital aspects of design refinement, so that errors get caught early in the process, which results in reduced time-waste. Usability testing is executed to incorporate feedback into the process. In this step, a prototype's usability and functionality are evaluated by making users go through a series of defined tasks [8]. Depending on the project, the testing can be conducted on paper prototypes, an interactive prototype, or the product itself.

The scale of the testing also varies depending on the project's budget, schedule limitations, and resources [20]. When the usability of a prototype is tested, the feedback can be used to check for improvements which can be handled in the next iteration. It is therefore beneficial to commence the user testing phase early in the process.

2.5 Agile Workflow and Iterations

Agile is a mindset and a way of working as a team. It is common in software development to incrementally deliver on small goals for a product. The team members assume responsibilities such as Product Owner and Scrum Master, and use a scrum board to monitor development as user stories are turned into tasks with ranging levels of difficulty. Stand-up meetings, which are shorter gatherings where team members can report on their progress, are led by the Scrum Master who also makes sure that the meeting stays on track [21].

Developing an interface is a process that continuously leaves room for improvement; in order to advance incrementally, iterations can be applied to the workflow. Each iteration has a new version of the product that can be tested to gain more feedback for the next iteration. A so-called Minimum Viable Product (MVP) should be the first goal, and provide basic features at bare minimum that can generate constructive feedback from customers [20]. The purpose of this is to leave the details to a later stage of development and create a strong foundation first. It can reduce the amount of work that needs to be readjusted, and having an MVP will therefore optimise time efficiency and resources.

3
Technology Choices

This section provides a detailed overview of the technologies utilised in the development of the web application. The choices were made to ensure the smoothest possible development process, and each tool and library was selected for its unique features.

3.1 Language

The web application was coded in the language TypeScript which is a superset of JavaScript. TypeScript enhances the attributes of JavaScript but requires static typing which minimises the number of type errors while running the code and consequently makes the product more robust and maintainable [22].

3.2 Libraries

Libraries are building blocks that provide pre-written code which facilitate easier and more efficient software development. They enable developers to avoid low-level details by providing a means to add functionality without writing code from scratch [23]. The sections below details the various libraries integrated into the development of the web application.

React. React is a JavaScript library that also supports Typescript and is used to build user interfaces based on components. Due to React being a popular library for building user interfaces [24] and having a large set of libraries for additional functionality, the team decided to code the front-end with it. React's component-based architecture allows developers to create modular and reusable UI components, streamlining the development process and ensuring a consistent look and feel across the application. In addition, React provides libraries for testing the components and functionality of the application [25].

The decision to use React was strategic for this open-source project. Given React's popularity, it aligns with the goal of ensuring easier contribution from other developers.

Material UI. Material UI (MUI) is an open-source React component library which is used by many big companies [26]. MUI provides a collection of widgets and components to enhance the functionality and design of a front-end. The decision to

incorporate MUI into the project stemmed from its potential to reduce development time. By using MUI's component library, the team was able to focus on making the application even better rather than spending time on crafting common UI elements from scratch.

Axios. Axios is an HTTP library that enables users to make requests to the server to fetch data [27]. In this project, Axios was used to facilitate communication with the server through the API.

3.3 Tools

This subsection describes the tools employed in the development of the web application.

Docker. Docker is a platform used for developing, shipping, and running applications. Docker enables separation between application and infrastructure [28]. In this project, Docker was used for initiating the server-side environment. By containerising the server, Docker streamlined the process of setting up a consistent and isolated development environment, ensuring that all team members could work in a standardised setting.

Postman. Postman is an API development tool that is used to test API requests and responses [29]. In this project, Postman was used to facilitate the testing and validation of the API endpoints. Postman allowed the team to advance in the development process by enabling them to simulate user actions, such as creating and logging in a user, directly through the API. This approach saved time by allowing to test functionalities dependent on other states, even before everything was fully integrated into the application.

GitHub. GitHub offers a cloud-based Git repository that helps developers store, manage, track, and control changes to their code. It is a tool for coordinating work among different developers [30]. Since WebDP's repository was hosted on GitHub's platform, it was a natural choice to use it for this project. GitHub also offers a scrum board feature called *Projects*, where you can keep track of the progress of the project.

Figma. Figma is a collaborative interface design tool [31]. The decision to use Figma was easy, as many of the group members had already used it for making prototypes. Figma enabled the team to visualise and test user flows and interactions before any code was written.

Swagger. Swagger is a set of open-source tools that help developers to design, build, document and use REST APIs [32]. By utilising Swagger, the team was able to easily check the API's endpoints and specifications. This access provided a clear overview of the API's structure. Swagger's detailed documentation enabled the team to quickly understand and interact with the API, ensuring that the front-end implementation aligned with the back-end functionalities.

Google Chrome Lighthouse. Google Chrome Lighthouse is an open-source program used for evaluating and enhancing a web page's quality. It can analyse a website from various factors including performance and accessibility. The analytics are performed by comparing the web page with different best practices and standards. From this, the user is given a score, which is derived from a weighted average of all the tests that Lighthouse runs. The scores are represented on a scale from 0-100. In order to deliver a satisfactory user experience, websites should aim for a high score, which is above 90 [33]. By using Lighthouse as an evaluation tool, the group could quickly receive input on the quality of their page and to confirm that it met the requirements of accessibility. Many of the WCAG 2.1 Level AA [9] criteria are covered with Lighthouse's tests.

4
Methodology

The project's methodology was divided into several phases which required different approaches. In this chapter, the team's progress towards the end product will be presented in detail.

4.1 Researching Related Work

At the start of the project, the group was presented, by the supervisor, OpenDP's front-end called DPCreator [34]. The purpose of this was to decide whether to build upon this application or to start from scratch. A first step was therefore to do an analysis of the application in order to understand its functionalities, design, architecture, and how it handled differential privacy. This was done by trying to run their demo and study its design, functionality and architecture. The group also contacted the creators of DPCreator and arranged a meeting in order to receive help to run it and gain input on its performance. The creators indicated that DPCreator had a lot of problems due to not being worked on for about a year. These insights led the team to conclude that FrontDP should be built from scratch, taking some inspiration from the design of DPCreator. See Figure 4.1, Figure 4.2, and Figure 4.3, for selected parts of the DPCreator interface, that has been used as an inspiration for the front-end.

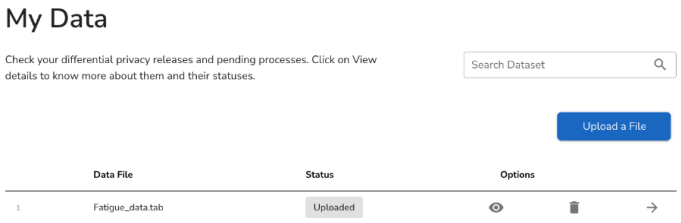


Figure 4.1: DPCreator: Data curator's list of datasets

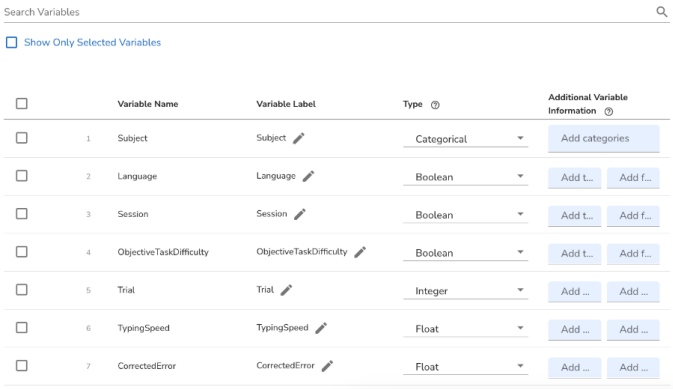


Figure 4.2: DPCreator: Data curator's variable selection

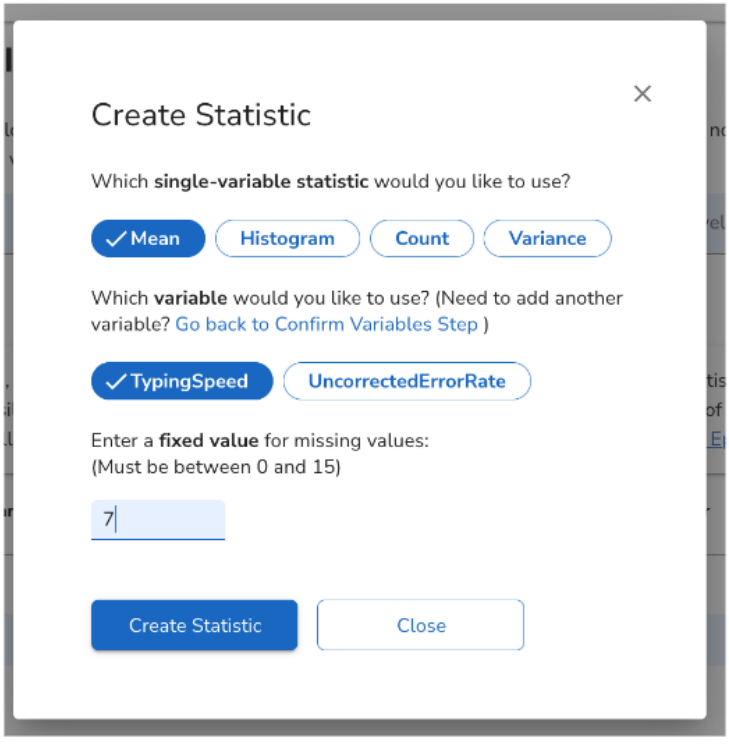


Figure 4.3: DPCreator: Data analyst's statistic selection

4.2 Prototyping

The completion of the research resulted in an educated vision of the goal, which was carried forward to the next step of the process: prototyping.

4.2.1 User Stories

Before the development of the prototype the group focused on creating user stories according to the given API. The user stories were divided into three focus areas. The first two focus areas were the main clients of the end product: the data curator and the data analyst. The third focus area included the admin user, as well as the general pages and functionality. The main goal of the creation of the user stories was to break them down into as small tasks as possible. See Appendix A for the initial user stories that were made. Later, the team used the user stories in order to create a scrumboard to get an overview over the project's progression. The scrumboard was made by using GitHub's management feature *Projects*, as seen in Appendix A.

4.2.2 Figma Prototype

After deciding on starting from scratch and creating user stories, the team began designing the prototype. After deciding to start from scratch, the team created user stories to define the tasks. Using these user stories as a guide, the team then began designing the prototype to fulfill the specified demands. The prototype was done in Figma and some of the inspiration of the layout was taken from the design of DPCreator and in accordance with the already mentioned reference guides [8][18]. See Appendix C for the Figma prototype.

4.3 Evaluating

Evaluation of the product and process was a central part throughout the project. Besides evaluating the product both within the group and individually, different sources were asked to give input on the product.

4.3.1 Feedback From DPella

One of the parties from which the group received feedback was the representatives of DPella. This was mainly done during the weekly meetings with Head of Business Development Carola Compá and the project's supervisor Alejandro Russo, who is the Co-Founder and CEO of DPella.

4.3.2 Usability Testing

When the interface design was finished, the group invited others to test the prototype. The decision of who would be asked to test the product was based on the person's previous experience with other similar applications. The group therefore requested the associates of DPella to give their opinion of the design. All of the

testers were handed a document with instructions on how the test would be performed. The instructions were planned with the goal of covering every user story of the application. In addition, when the person had completed the tutorial they had the opportunity to freely explore the prototype before writing feedback.

4.3.3 Analysing Accessibility

To evaluate and measure the accessibility of the web application, the group used Google Chrome's Lighthouse extension. Every time a page of the product was finished with the design and functionality, its accessibility was assessed with the help of the program. Even though Lighthouse offers a wide range of analytics metrics, the group decided to conduct the assessment using only the accessibility parameter.

5

Results

The final product, a front-end for WebDP, is presented in this chapter. This includes a detailed walk-through of the application, the design patterns that have been utilised, coverage of the API, the application's accessibility score, as well as the results of usability testing.

5.1 User Interface

This section provides a walk-through of the web application's user interfaces. It is presented from the perspectives of the three roles: admin, data curator, and data analyst.

5.1.1 Admin

The home screen for admin users is a page where they can manage system users, as shown in Figure 5.1. The admins can add new users by clicking the *Add* button at the bottom of the list. When the *Add* button is clicked, a new row will appear in the list where they can enter the new user's details, as shown in Figure 5.2. Once complete, admins can click *Confirm* to add the user or *Cancel* to abandon the process. If there are fields that are not filled in, or if they attempt to create a user with a username that is already taken, a message will pop up to inform the admin.

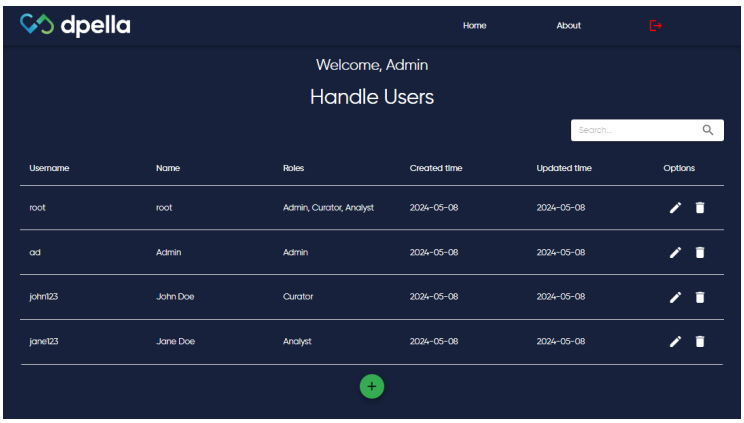


Figure 5.1: Admin's home screen

The admin can also delete users by clicking on the trash can icon. A confirmation dialog will appear to ensure that the admin is sure about the deletion. Additionally,

they can edit a user by clicking on the pen icon, see Figure 5.2. After clicking, they can modify the user's name and role. Admins can then confirm the edits by clicking on the checkbox or cancel by clicking on the cross.

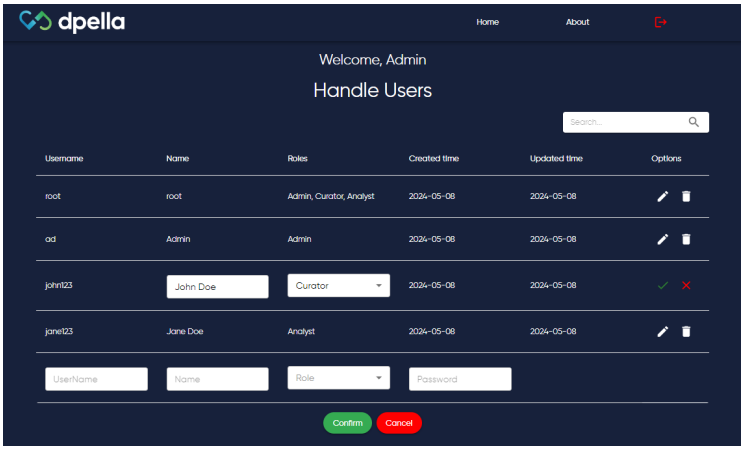


Figure 5.2: Adding and editing user on admin's home screen

Several design patterns are utilised on this page in order to follow design principles and enhance user-friendliness. For example, there is a visual framework displayed on this page but also on every other page of the application. The framework consists of specific font, colors, shapes and similar, which results in the different pages of the application being coherent. The pattern called escape hatch is also utilised and implemented through the company logo, so that the user can return to the home page without inconvenience. The prominent done button pattern is used and is displayed when the admin adds or edits a user. See the button for confirmation in Figure 5.2. Other design patterns that are displayed in the figures that represent the admin screens are sign-in tools, input hints, and drop-down list.

5.1.2 Data Curator

The data curator's way through the application is illustrated in the flow diagram represented in Figure 5.3.

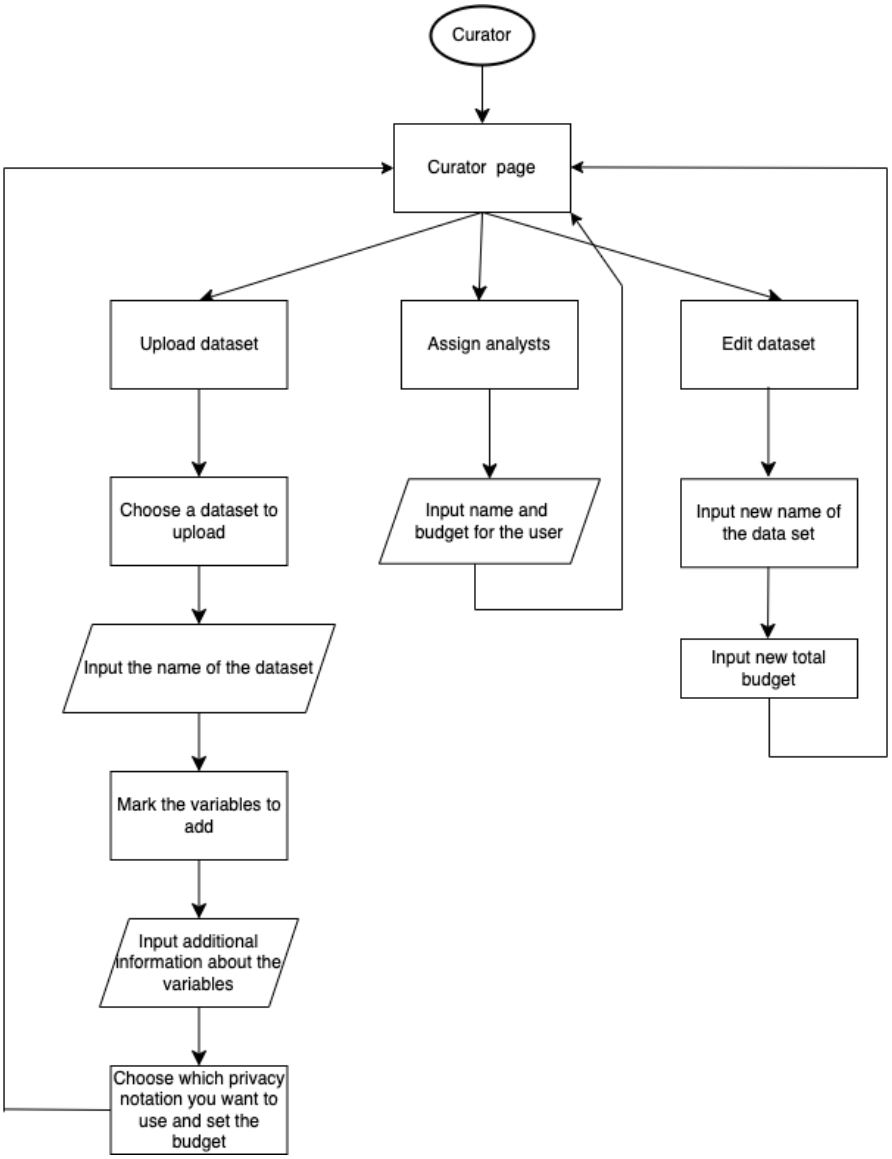


Figure 5.3: User flow diagram data curator

After a user has logged in with the role of a data curator, it enters the home screen of its role, as shown in Figure 5.4. On this page, the user can start a new upload or manage previously uploaded datasets. If no dataset has been uploaded yet, the curator will only be able to upload a new one.

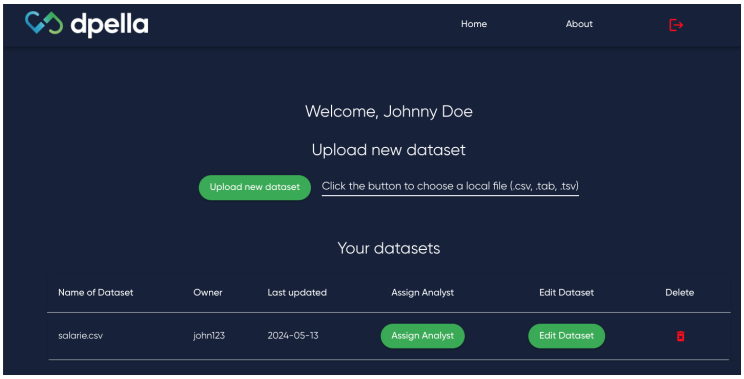


Figure 5.4: Data Curator's home screen

After choosing a file to be uploaded, the user is directed to a multi-step form where certain settings can be set. First, the user is given the choice of renaming its dataset, as seen in Figure 5.5. The wizard design pattern is implemented in the uploading form. This provides an overview of the uploading process, so that the user gets a comprehension for where they are in the process, and what the previous and next steps are.

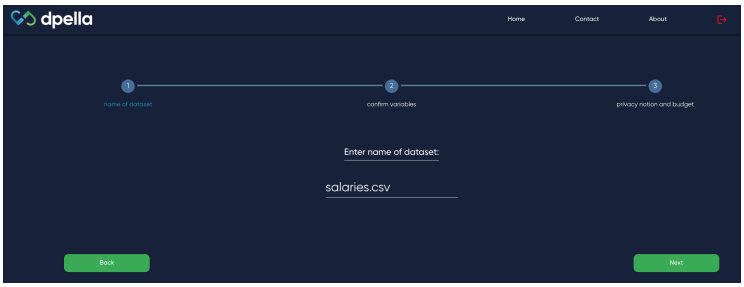


Figure 5.5: Step one in the upload process of the dataset

Afterwards, the user will be able to select which of the variables in the file that the dataset should be analysed on. In addition, the curator can specify a type for every variable, namely Text, Enum, Integer or Double. Depending on the type, the curator can set different boundaries to the analytics of the selected variable, as illustrated in Figure 5.6.

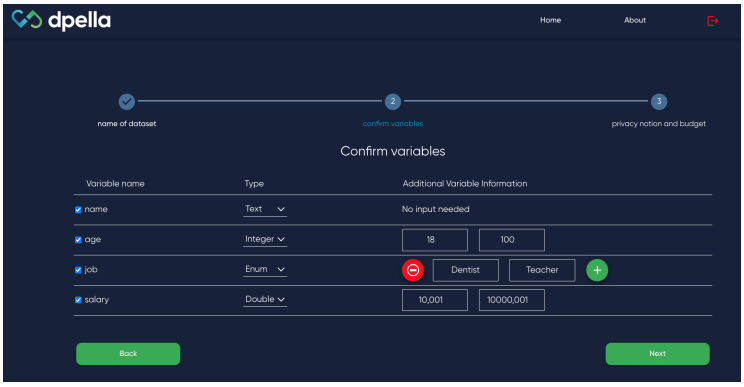


Figure 5.6: Step two of the upload process of a dataset

Lastly, the user must assign a total budget for the dataset, as shown in Figure 5.7. The user must always enter an epsilon value. Epsilon is an upper bound for the amount of privacy lost, with a maximum probability delta that the loss will be greater. If the curator chooses the option *ApproxDP* as the privacy notion, they also have to enter a delta value as input.

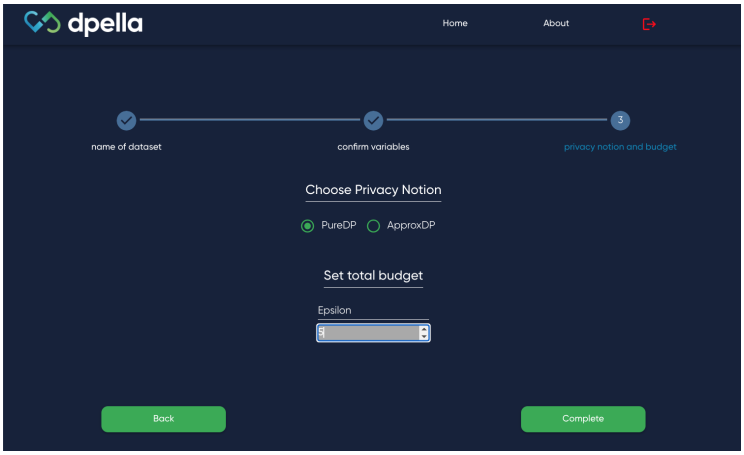


Figure 5.7: Step three of the upload process of the dataset

During the whole form, the user will receive notifications when proceeding to the next step if there is any information missing or non-allowed inputs. When the user is finished and presses *Complete*, the settings of the dataset will be uploaded as well as its data.

After uploading a dataset, the curator will be redirected to the home screen again, where they can apply different settings to its uploaded datasets. The curator can choose between assigning analysts, edit the dataset's budget, and name or delete one of their uploaded datasets. If the button *Assign analysts* is pressed, the curator will be directed to another page where all of the available data analysts will be listed. Here, the user can assign one or more analyst to the dataset and give them an individual budget. This page is shown in Figure 5.8. On this page, the design pattern drop-down selector is utilised in order to clarify to the user which analysts

that can be assigned to the dataset.

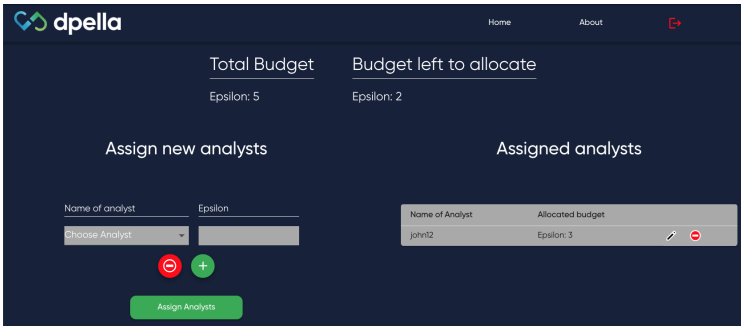


Figure 5.8: Page for assigning new analysts to datasets

If the curator presses *Edit Dataset*, they will instead be directed to a page where they can change the dataset's name or update it with a new total budget. This is implemented with a multi-step form similar to the form in the uploading process, as illustrated in Figure 5.9 and 5.10

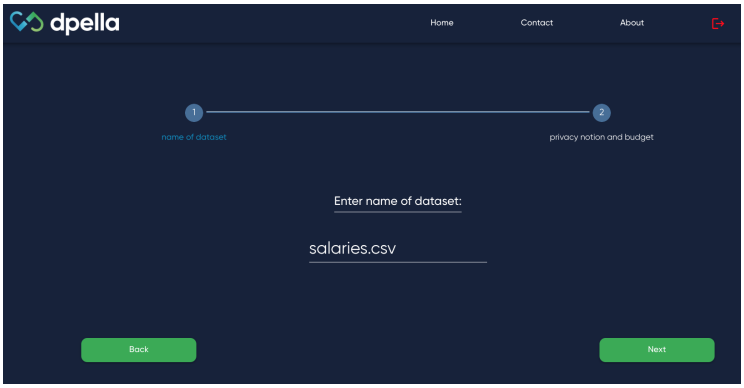


Figure 5.9: Step two of the editing process of the dataset

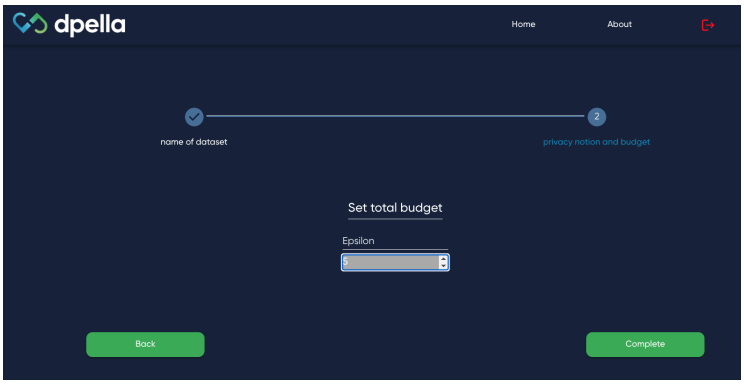


Figure 5.10: Step two of the editing process of the dataset

5.1.3 Data Analyst

Figure 5.11 below presents the user flow diagram for the data analyst role. It serves as a visual guide to the tasks analysts can perform within the application.

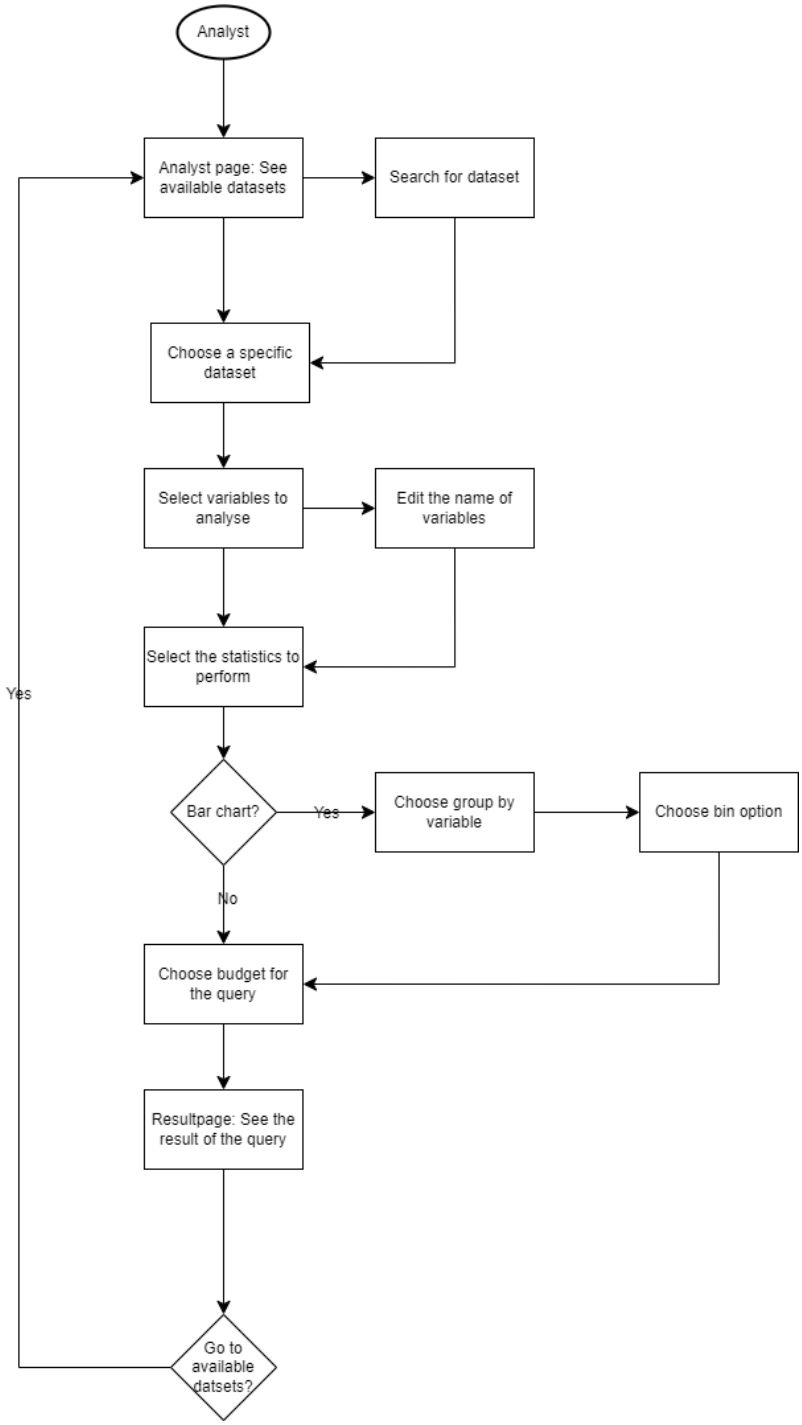


Figure 5.11: User flow diagram data analyst

The initial screen for a data analyst displays a list of datasets assigned to them, as shown in Figure 5.12. Analysts can view details about each dataset and use a search bar to quickly find specific datasets. Each dataset includes an options column, where analysts can select to continue with a dataset. Here, a grid view is used so that it is clear what datasets are available, and what the specifications of each dataset are.

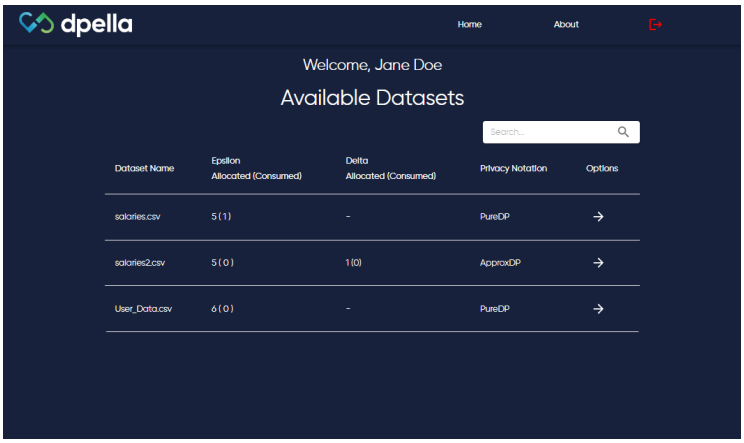


Figure 5.12: Data analyst's home screen

When the analyst chooses to continue with a dataset, a multi-step form is displayed. The chosen dataset and the remaining budget are shown at the top of the page to ensure that the user feels confident about what dataset they are handling.

The first step in the form requires the user to mark the variables they want to analyse, as shown in Figure 5.13. The variables displayed are selected by the data curator and are listed with their respective details. If the user wants to assign different names to the variables, they can edit the variable labels by clicking on the edit icon. Several of the design patterns in this page are used in pages shown in previous figures, since they serve the same purpose for this part of the web-application as well. For example, the multi-step form is visualised with a wizard to provide an overview of the steps. There is also a combination of list and grid view that visually clarifies the variables.

The next step in the form is where analysts choose the statistics they want to perform, as shown in Figure 5.14. The options are mean, sum, minimum, maximum, and count, and the user is only allowed to select one. This is made clear with the implementation of the toggle button group design pattern. Then, the user must choose a variable to apply the selected statistic on. Here, too, only one choice is permitted. The user can now continue to the next step, or choose the bar chart option, visualised as a toggle button.

If the bar chart option is selected, a *group by* option appears, as shown in Figure 5.16. Here, the user can choose how to group the bars in the chart. Upon selecting a *group by* variable, the question *Choose option for bins* is displayed. If the *group by* variable is numeric, the user can choose between two radio buttons that repre-

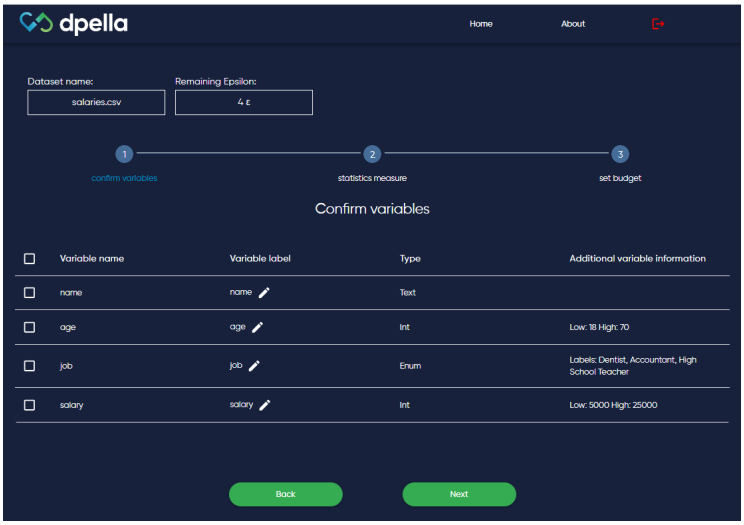


Figure 5.13: Continue with dataset

sent *One bin per value* or *Equal range bins within variable bounds*. If the latter is selected, a slider is displayed where the user can set the number of bins. If the *group by* variable is of type Enum, the corresponding labels are displayed as bin options.

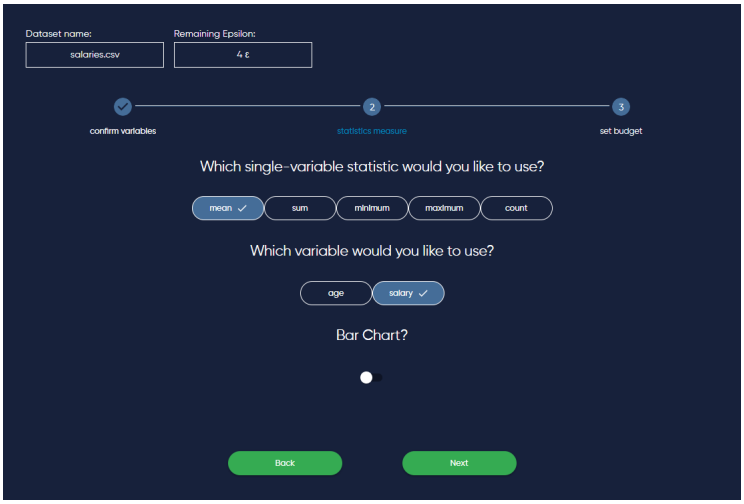


Figure 5.14: Choose statistics

The last step in the multi-step form is to set the budget for the statistics, as shown in Figure 5.15. If the budget includes both epsilon and delta, a slider will be displayed for each. After selecting the budget, the next step is to create the statistics by clicking on the *Create Statistics* button.

Creating the statistics can take some time, so while the user waits, the *Create Statistics* button changes to a circular progress visualisation. If the user has opted to perform statistics without the bar chart option, the result is displayed in text, as shown in Figure 5.17. If the bar chart option is chosen, the result is displayed as a bar chart, as shown in Figure 5.18.

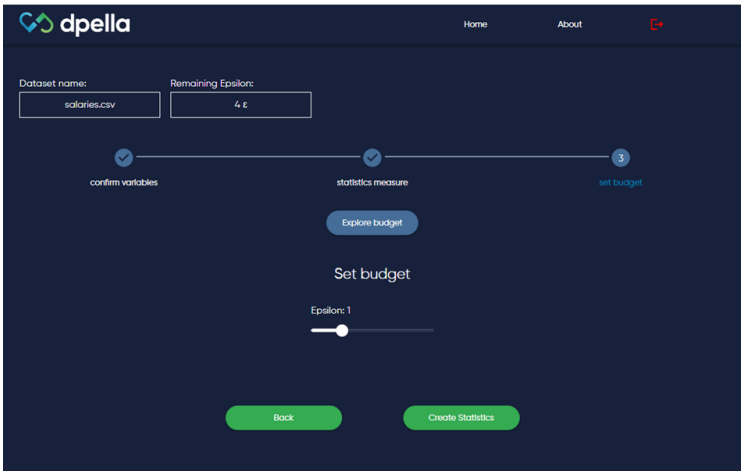


Figure 5.15: Set budget

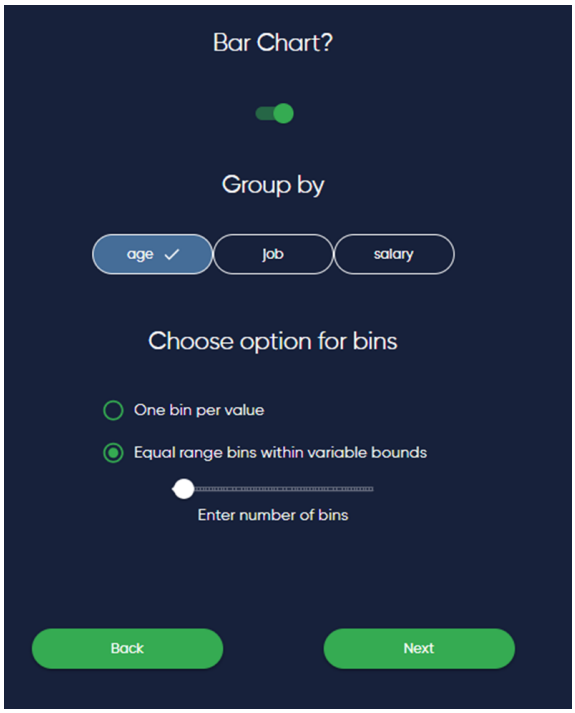


Figure 5.16: Bar chart chosen

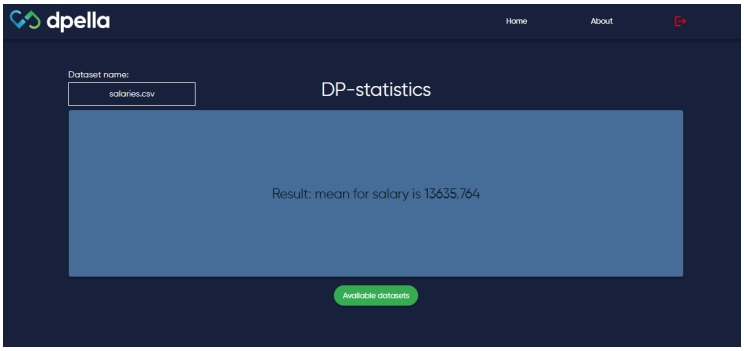


Figure 5.17: Result without bar chart

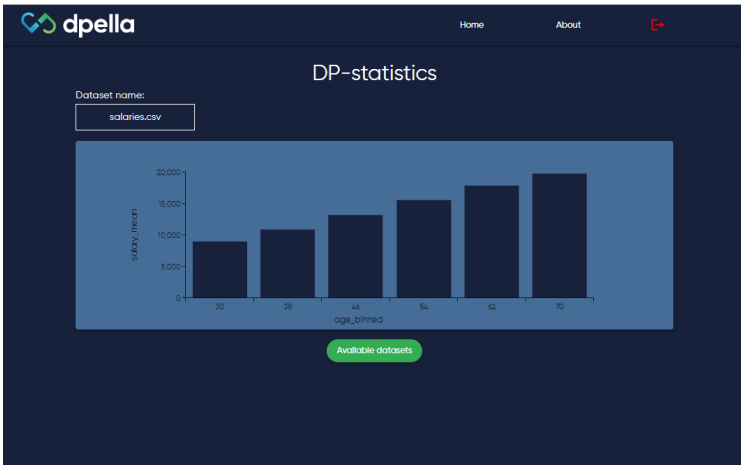


Figure 5.18: Result with bar chart

5.1.4 General Pages

Besides the role specific pages, all users have common pages which are not dependent on any role. One of them is the starting page which users are navigated to when entering the website. It is shown in the Figure 5.19. The user have the options of either navigating with the buttons on the navigation bar or to login to its account with the login button. The user is then directed to the page illustrated in Figure 5.20. This page is also the one the user is directed to when they log out from their account, which is done by the log-out button at the top right in the navigation bar.

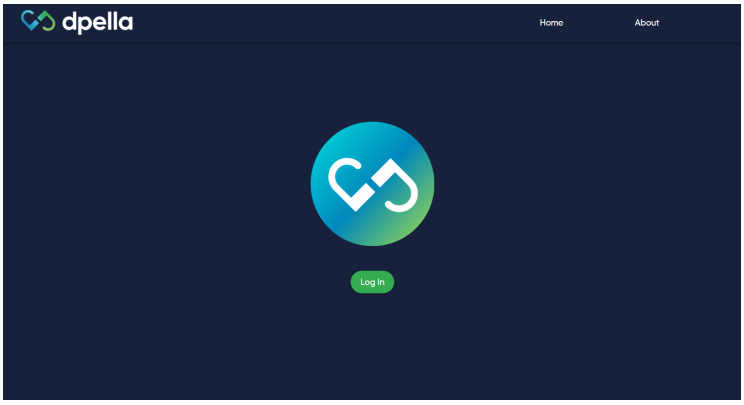


Figure 5.19: The starting page

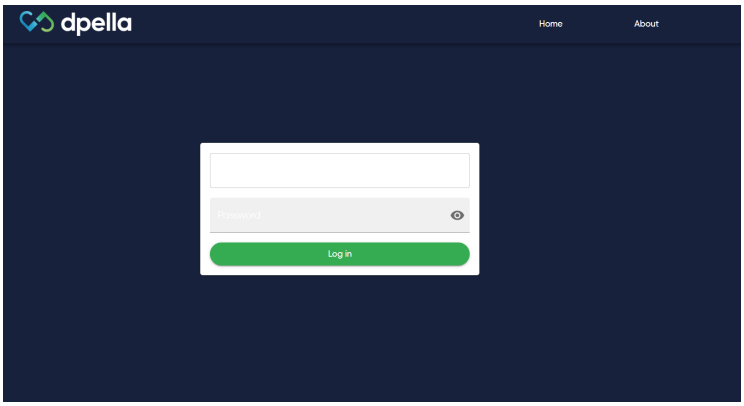


Figure 5.20: The log-in page

In addition, all users can navigate to two pages which are dedicated to DPella and DP. They are accessed through the navigation bar via the *About* button. When pressed, the user has two options: to either visit *About Us* or *About DP*. In the former, the user can read information about DPella and their associates. This is illustrated in the Figure 5.21. In the latter, a comprehensive text about DP is presented for the user, which is shown in Figure 5.22

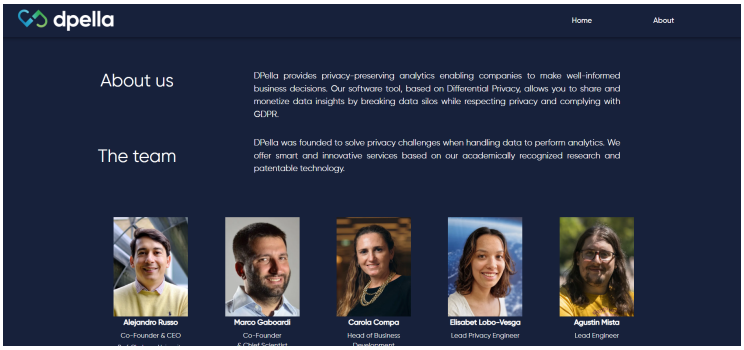


Figure 5.21: Page with information about DPella

Figure 5.22: Page with information about DP

5.2 API coverage

One of the main goals of this project was to cover as many parts of the API as possible. In the end, the application utilised 20 of the 22 API endpoints. The two endpoints that were not implemented are the enabling of custom queries, and computing the accuracy of a query. The former was not included in this version because it introduces complexities in design, impacting both security and ease of use for the user. The latter was implemented but not used, as the only engine in the current version, Tumult, did not support this type of calculation.

The application makes use of most features of the evaluate query endpoint. However, two functionalities are not implemented: filtering data and mapping data. Filtering data, which for example allows users to consider only ages within certain limits, was not implemented due to a lack of time and some technical issues with the Tumult engine that handles these queries. Mapping data, which involves the aggregation of two variables, was also not prioritised due to time constraints. Furthermore, the variable type 'boolean' was not implemented in the application because Tumult does not support booleans in the evaluate query endpoint.

5.3 Accessibility

The accessibility was assessed throughout the project with Google Chrome's Lighthouse extension. The result is presented in the Table 5.1. In the table is the different pages divided into three different categories - Data Curator, Data Analyst, and General Pages including Admin, in order to give an overview of how well the different parts of the front-end perform in the accessibility aspect. Lastly, the average score of the whole product is presented. The most common improvement opportunities that would increase the score were to increase the contrast for some background and forefront colors, set the headers in a more sequentially-descending order and add aria-labels to input forms. The aria-labels provide text labels for objects, allowing screen readers to inform blind users what the objects are.

Type of Page	Average Score
Data Curator	81
Data Analyst	89
General Pages	90
Total Average	87

Table 5.1: Average accessibility scores on the different parts of the program

5.4 Usability Test

The usability test document was created and sent to associates of DPella, who individually tested the design in Figma. This generated a response as seen in Appendix B. The feedback was sent by Elisabet Lobo-Vesga, Lead Privacy Engineer at DPella, and agreed with by Carola Compa who shared similar thoughts. Aside from specific feedback on Figma functionalities and typos, the team received useful feedback on design details and logic; for example, to clarify allocated and remaining epsilon and delta values in the Data Analyst Screen, which the team changed in the implementation, as seen in Figure 5.12.

6

Discussion

In the preceding section, the final product was presented: a web application designed to facilitate data analysis with DP. The application is tailored for three roles: admin, data curator, and data analyst. For admins, the system provides user management capabilities, including the addition, deletion, and editing of user accounts. Data curators have access to an interface for uploading, configuring, and editing datasets, as well as assigning data analysts. Meanwhile, data analysts are equipped with tools for conducting statistical analyses on the assigned datasets. This section will discuss how the group came to these results.

6.1 The Development Process

The development process consisted of several steps. In this subsection, these different parts of the process will be evaluated.

6.1.1 Studying Related Work

As previously mentioned, the group was presented OpenDP's front-end DPCreator at the beginning of the project. A choice had to be made whether to build upon the existing product or to start from scratch. It was a pivotal moment when the group decided on the latter, since it affected the project to great extent. Due to the fact that the group's decision was taken five weeks into the project and therefore somewhat delayed, the programming of the product had to be done fairly quickly. On the one hand, this decision might have affected the end product negatively because of the time constraint which followed. For instance, the group could have been able to cover more of the API's functionality or evolve the design even further. Instead, time was spent on laying the ground work that comes with starting from the beginning, for example setting up the environment and deciding the structure of the code. On the other hand, building upon an established product is also time consuming, especially since no one in the group had any prior experience with the programming language and framework Vue, which DPCreator was built upon. Another important aspect which grounded the decision even more was that DPCreator was built with a prior version of Vue, namely Vue 2. As a result, many of the product's libraries were deprecated and therefore caused dependency errors in the functionality.

6.1.2 Collaboration in the Team

Different tools for collaboration have been presented throughout the bachelor's programme and they proved to be really effective during this project. Figma allowed the group to collaborate with the design in an effective way. This helped the group to visualise together and present to DPella a possible user interface of the end product in the early stages of the project. Consequently, this allowed the team to get approval on the design before the actual implementation started.

Additionally, the agile process of working with the implementation resulted in group members effectively building different aspects of the front-end and to shine in their respective areas. Through the help of Postman and GitHub each member could work on their respective area without having the other areas of the program completed.

6.1.3 Defining Requirements, Scope and Users

Due to time constraints after deciding to build a front-end from scratch, the requirements and scope of the project were limited to a set of tasks that the group could realistically complete within the remaining time frame. Two major questions arose: "What is going to be implemented?" and "Who are the users going to be?". With the help of the supervisor, the team gained a clear understanding of what DPella wanted most for the product, what should be covered in the API, and who the potential users were. DPella found the logic of uploading, editing and querying the datasets to be the most important parts of the application. This was therefore prioritised during the development.

6.1.4 Designing the Interface

Based on the information provided by the supervisor regarding what the interface should be and who the intended users are, the group began designing the front-end using Figma, taking DPella's wishes into consideration. The designing process began immediately in Figma which deviated from the norm since one is supposed to create a basic sketch prototype first. A basic sketch could have resulted in a more common view of the design, which would have been more effective. However, while researching DPCreator, the group decided to use its design as an inspiration and saw it as a replacement for the basic sketch.

An important aspect of prototyping is to get valuable usability testing experience before beginning the implementation. The reasoning behind this is efficiency, as attention can be entirely focused on implementing a design once it has been accepted. The initial thought was to schedule usability testing with students from the IT program at Chalmers in order to gather large amounts of feedback. However, due to time constraints the team had to compromise and instead only invited the associates of DPella to give feedback on the design. The team deemed Chalmers students to be not as optimal users for conducting usability tests because of their inexperience within the subject.

The DPella team tested the prototype and gave feedback at the tail-end of the project. Although the process of usability testing was delayed, the group found the user-testing aspect of prototyping invaluable. This is due to the fact that it is difficult to simulate average users and getting feedback from them could be worthwhile to really make sure that the design is appropriate. Furthermore, in a similar future project, this aspect should not be compromised on.

6.1.5 Implementation

During development, the team encountered some challenges with the engine. In the current version, the only engine used is Tumult, which does not support all of the functionality in the API. This caused some time loss because some UI elements were implemented before realising that they were not supported by Tumult.

Furthermore, during the implementation phase of the project, significant time constraints impacted the ability to conduct comprehensive automated testing. Instead, manual testing was employed to find bugs. This approach allowed for the quick identification and resolution of issues as they arose, providing a practical solution under the tight deadline.

6.2 End Product

This section will discuss the result of the end product and present arguments for it. In addition, possible future work will be presented at the end. Evaluating whether a design is good or not is a non-trivial task since there are many aspects to a design. Whether something has a pleasant appearance or not is mostly subjective, so the design will be evaluated against the goals the group had set in the beginning. By adhering to well-established design patterns and commonly used principles, the team built a well-thought out design for the front-end.

6.2.1 Arguments for Design Decisions

The colour scheme and logos were provided by DPella, which allowed the group to create a design that followed the company's theme. All buttons have clear wording, so the action of pressing a button does not surprise the user. Additionally, most of the buttons change colour on hover. Each aspect of either uploading, assigning or running a query has separation to make each step clear and understandable. In addition, both the curator and analyst has a multi-step form with a wizard when handling larger processes, to not overwhelm the user.

When presented with DPCreator, the team not only had to determine the feasibility of developing it further, but also to take notes on the design to use as inspiration for their own design. The DPCreator team made some design decisions that render their front-end simple to use and comprehensible. The team, in their exploration of DPCreator, therefore drew some inspiration from their design. The list of various

uploaded datasets, which is available to data curators and analysts, is one feature that the team particularly liked. The team decided to include extra details about the datasets in their list, so the list headings are not exactly the same. The variable selection for the query and dataset upload process was another thing for which the team took inspiration from DPCreator. When confirming variables, the user is presented with a lot of information, so the team consequently decided that the best way to display the information would be in a list similar to that of DPCreator's. For comparison, see Figure 4.1 which displays DPCreator's curator screen, and the final screen of FrontDP in Figure 5.4.

The data curator screen having a dashboard design allows the curator to easily access and upload new datasets. In the prototype, assignment of analysts was done at the end of the multi-step form. However, in the end product, this functionality is placed at the home screen of the data curator. This decision was made so that the curator gets full control over each aspect of their dataset and to simplify the process of uploading and adjusting it. For the steps of uploading the dataset, inspiration was drawn from DPCreator in how the different variables are displayed, specifically when selecting which variables to allow the analyst to run queries on, as seen in Figure 4.2.

The home screen for data analysts is inspired by the DPCreator. It features a similar clean design, including a list of assigned datasets, a search bar, and a title. The arrow button in the list, which allows the user to continue with a specific dataset, is taken from DPCreator. The structure used in DPCreator for continuing with a dataset has also inspired the design. The first step in the multi-step process of DPCreator, where users can select the desired variables, has been adopted in the design. However, the next step in DPCreator's design, which was found to be a bit confusing, involves selecting a budget and clicking a 'Create New Statistics' button. This process has been divided into two separate steps. Instead of a single 'Create New Statistics' button, the query component is now a distinct step. In DPCreator, clicking 'Create New Statistics' triggers a pop-up with several options, as seen in Figure 4.3. Inspiration for the query step was drawn from these questions and buttons, but the design was simplified to enhance clarity and ease of use for the user.

6.2.2 Considerations on User-Friendliness

The group had a goal in mind: to create a user-friendly design characterised by simplicity in navigation and enabling safe exploration of the service. Some aspects considered included ensuring that users would never be confused. Key decisions during the development process, such as employing the principle of separation of concern, have led the group to feel that this goal has been achieved. Each step is as simple as could be, but an important note is that the design still requires the user to have some knowledge about DP. For instance, tutorials outlining what will occur at each step may be added. However, after talking with the supervisor, the group decided that since a reasonably informed user is what is expected of the service, it would not be wise to add features that might disrespect the user.

Another important aspect when it comes to user-friendliness is accessibility. If a front-end has high accessibility, it allows a greater variety of people to use it. It could for instance be more accessible for a greater number of individuals with disabilities, encompassing modifications for low vision and blindness, hearing loss and deafness, restricted mobility, speech impairment, and photo sensitivity. The result was measured by using Google Chrome's extension Lighthouse in order to assess every page. The assessments from Lighthouse gave us as developers insight into how well the product followed general best practices and official guidelines, such as the WCAG. The group knew that a score of at least 90 was classified as sufficient and anything below as substandard. However, in the end the average score of the application's pages was just below the approved limit with a average score of 87. In order to improve the result, the product should for instance be implemented with aria-labels to all input fields, headers should be in a sequentially-descending order, and there should be more contrasts between some background and forefront colors. This was not implemented due to time constraints since other parts of the project were more prioritised, such as fulfilling as much of the API's endpoints as possible. However, the group found this somewhat problematic since this is an aspect which is probably often neglected when performance is mainly in focus and prioritised. Consequently, it results in an even greater challenge for people with disabilities to use the products and be engaged in society.

6.2.3 Future Work

Along the way, many ideas for additional features emerged that could make the application even better in the future. One of the API endpoints the team did not have time to implement was the custom query endpoint, which requires only a string input. One idea for this implementation could be to add an advanced option when the analyst is choosing statistics. Once the user selects the advanced option, they would be able to enter their query in a text field. This approach would offer analysts greater flexibility and enable them to perform more complex queries. Implementing this feature could be advantageous as it would allow users, regardless of their query knowledge, to utilise the application efficiently.

Another endpoint that was not covered relates to calculating the accuracy of a specific query. The idea for this involved adding an "explore budget" button at the data analysts' step when choosing the budget for a query. Upon clicking the button, a pop-up would appear where the user could select a budget and a confidence interval, and then the accuracy for the chosen values could be calculated. This feature would have allowed data analysts to experiment with how different budgets affect accuracy, enabling them to use their budgets wisely. Unfortunately, the design and implementation of this function were completed before it was realised that the Tumult engine does not support this feature. However, this approach would be suitable for implementation in the future.

One more feature for the future is to enable data analysts to filter the data to be

analysed. When the time came to implement this feature, a bug was discovered that was very time-consuming, which prevented full implementation. The idea was to have a toggle button, similar to the bar chart option, allowing the user to decide whether or not to apply a filter. The filter would have been placed beneath the variable selection, with sliders for numerical limits, and drop-down menus for the variable to filter and the comparison operators. Furthermore, the application did not utilise the text variable type. If the filter feature is implemented, text variables could also be used in the statistics.

Additionally, the group initially wanted to implement functionality that would allow users to save the queries they have performed on a dataset, along with their results. This feature would have been very valuable for users, allowing them to revisit certain results without having to expend more of their budget to repeat the query. However, the team quickly realised that this would require additional time since it was not part of the existing API, meaning the team would need to store the information elsewhere. Nevertheless, the query history functionality would be an important step if the product is developed further in the future.

Another potential future feature could be to add automatic recognition of additional information for the variables. If the dataset is large, it could be challenging for the data curator to know, for example, the minimum and maximum values of a variable. This feature would pre-fill the information but would also allow the data curator to edit it. In the current version, the data curator can enter a value that is, for example, much larger than the largest actual value. This leads to inaccurate query evaluation results for non-existent values.

6.2.4 Social and Ethical Aspects

The goal of DP is to protect individuals' data from being exposed. Personal information about an individual is very valuable and, depending on what it contains, may have detrimental social effects if disclosed. Public awareness of the individuals' right to data privacy is growing, and so are the demands on companies to handle the data responsibly. The team's contribution to an open-source project such as this one raises awareness of DP and facilitates the adoption of safer data practices on a larger scale.

Moreover, while increasing the awareness of DP, one should simultaneously prioritise the development of accessible user interfaces. Focusing on implementations such as keyboard navigation, colour-blind-friendly palettes, and alternate text for pictures, results in a product that is more user-friendly for a wider range of users' demands and abilities. This imperative aspect of front-end design adheres to the ethical principles of inclusiveness, expanding the product's potential market.

7
Conclusion

The purpose of the project was to build a front-end that made DP easy and accessible to both data analysts and curators. The front-end was built in React to enable further development and maintenance, which is crucial in order to ensure the project's longevity. The project focused on the design to make it user friendly and comprehensible. Nearly the entire API provided by WebDP was covered in the final product. Data curators can upload datasets, set corresponding limitations, and assign analysts to the datasets. The assigned analysts can run statistical queries on the datasets. Future work could include functionality for the users to save past queries, and to further develop unimplemented API functionalities.

Bibliography

[1] D. Kifer, S. Messing, A. Roth, A. Thakurta, and D. Zhang, *Guidelines for Implementing and Auditing Differentially Private Systems*, 2020. arXiv: 2002.04049 [cs.CR].

[2] F. McSherry, "Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis," *Commun. ACM*, vol. 53, no. 9, p. 89, Sep. 2010. DOI: 10.1145/1810891.1810916.

[3] E. Lobo-Vesga, A. Russo, and M. Gaboardi, "A Programming Framework for Differential Privacy with Accuracy Concentration Bounds," *2020 IEEE Symposium on Security and Privacy (SP)*, vol. 1, pp. 411–428, 2020. DOI: 10.1109/SP40000.2020.00086.

[4] *DPella*, DPella, [Online]. Available: https://www.dpella.io/ (Accessed: 8 May 2024).

[5] *Welcome to OpenDP*, OpenDP, [Online]. Available: https://docs.opendp.org/en/stable/index.html (Accessed: 8 May 2024).

[6] *Tumult Analytics*, Tumult Labs, [Online]. Available: https://tmlt.dev (Accessed: 7 May 2024).

[7] *What is RESTful API? - RESTful API explained - AWS*, Amazon Web Services, [Online]. Available: https://aws.amazon.com/what-is/restful-api/ (Accessed: 8 May 2024).

[8] A. Cooper, R. Reimann, and D. Cronin, *About Face: The Essentials of Interaction Design*. Indianapolis: John Wiley & Sons, 2014.

[9] *WGAC*, The World Wide Web Consortium, [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/ (Accessed: 21 February 2024).

[10] K. Missman, *Front-end vs. Back-end developers: What's the Difference?* Forbes, [Online]. Available: https://www.forbes.com/advisor/education/it-and-tech/front-end-vs-back-end-developer/ (Accessed: 11 April 2024).

[11] D. Nobelius and L. Trygg, "Stop Ahasing the Front End Process – Management of the Early Phases in Product Development Projects," *International Journal of Project Management*, vol. 20, no. 5, pp. 331–340, Jul. 2002. DOI: 10.1016/s0263-7863(01)00030-8.

[12] S. Lauesen, *Software Requirements Styles and Techniques*. London: Addison-Wesley, 2002, ISBN: 9780201745702.

[13] M. K. Fageha and A. A. Aibinu, "Managing Project Scope Definition to Improve Stakeholders' Participation and Enhance Project Outcome," *Procedia - Social and Behavioral Sciences*, vol. 74, pp. 154–164, Mar. 2013. DOI: 10.1016/j.sbspro.2013.03.038.

[14] J. Pruitt and J. Grudin, "Personas," *Proceedings of the 2003 conference on Designing for user experiences*, Jun. 2003. DOI: 10.1145/997078.997089.

[15] M. Godbolt, *Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites*. Sebastopol, CA: Oreilly Media, Inc., 2016, ISBN: 9781491926789.

[16] B. Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*. Amsterdam: Morgan Kaufmann, Dec. 2011, ISBN: 9780123740373.

[17] J. Johnson, *Designing With the Mind in Mind – Simple Guide to Understanding User Interface Design Guidelines*. Cambridge: Morgan Kaufmann, 2021, ISBN: 9780124079144.

[18] J. Tidwell, *Designing Interfaces*. Farnham: O'REILLY, Dec. 2010, ISBN: 9781449302832.

[19] K. Schleifer, *Mastering Frontend Testing: Essential Best Practices*, Medium, [Online]. Available: https://medium.com/comsystoreply/mastering-frontend-testing-essential-best-practices-b0924655e765 (Accessed: 7 May 2024).

[20] H. Sharp, J. Preece, and Y. Rogers, *Interaction Design: Beyond Human-Computer Interaction*. Indianapolis: John Wiley & Sons, 2019, ISBN: 9781119547358.

[21] K. Schwaber and J. Sutherland, *2020 Scrum Guide*, Scrum, [Online]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf [PDF] (Accessed: 8 May 2024).

[22] A. Pungu, *How Typescript Can Improve Your Web Development Projects*, freeCodeCamp, [Online]. Available: https://www.freecodecamp.org/news/how-typescript-can-improve-web-development-projects/ (Accessed: 8 May 2024).

[23] N. Gupta, *Frameworks vs. Libraries: What's the Difference?* Medium, [Online]. Available: https://medium.com/@namogupta208002/frameworks-vs-libraries-whats-the-difference-af1fa3df89c6 (Accessed: 11 April 2024).

[24] L. S. Vailshery, *Most Used Web Frameworks Among Developers 2023*, Statista, [Online]. Available: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (Accessed: 15 April 2024).

[25] *React – A JavaScript Library for Building User Interfaces*, React, [Online]. Available: https://legacy.reactjs.org/ (Accessed: 25 April 2024).

[26] *Material-ui - Market Share, Competitor Insights in Frontend Framework*, 6sense, [Online]. Available: https://6sense.com/tech/front-end-framework/materialui-market-share (Accessed: 15 April 2024).

[27] *Getting Started*, Axios, [Online]. Available: https://axios-http.com/docs/intro Accessed: 8 May 2024).

[28] *"Docker Overview"*, Docker Docs, [Online]. Available: https://docs.docker.com/get-started/overview/ (Accessed: 22 March 2024).

[29] *What is Postman?* Postman, [Online]. Available: https://www.postman.com/product/what-is-postman/ (Accessed: 8 May 2024).

[30] *About GitHub and Git*, GitHub Docs, [Online]. Available: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git (Accessed: 8 May 2024).

[31] *What is Figma?* Figma, [Online]. Available: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma (Accessed: 8 May 2024).

[32] *What is Swagger*, Swagger, [Online]. Available: https://swagger.io/docs/specification/2-0/what-is-swagger/ (Accessed: 8 May 2024).

[33] *Lighthouse Performance Scoring*, Google, [Online]. Available: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring (Accessed: 29 April 2024).

[34] *DP Creator*, OpenDP, [Online]. Available: https://docs.opendp.org/en/v0.2.1/opendp-commons/dpcreator.html (Accessed: 8 May 2024).

Appendix A

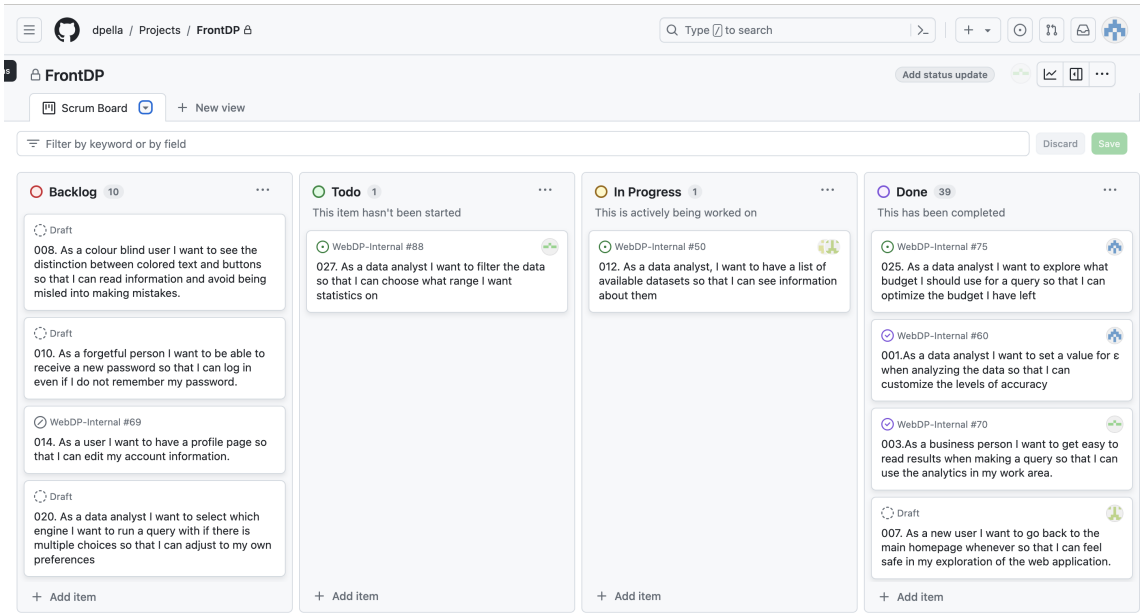


Figure 1: Scrum Board

DATX11-VT24-28

Andres Arriagada Fuentes,
Ann Heijkenskjöld, Kristin Hulling,
Eric Källman, Wilma Nordlund

DP- User Stories

User ID.	As a <...>	I want to <...>	So that I can <...>
001	Data Analyst	Set a value for ε when analysing the data	Customise the levels of accuracy
002	Uninformed user	Know about DP and benefits	Use the web application
003	Business Person	Get easy to read results when making a query	Use the analytics in my work area.
004	Data Curator	Be able to upload datasets	Assign the dataset to the data analysts
005	User	To be able to log in with my specific role	Do tasks that are available to me
006	User	To be able to log out	Be sure that no one else can use my account
007	New User	Go back to the main homepage whenever	Feel safe in my exploration of the web application
008	Colour blind user	See the distinction between colored text and buttons	Read information and avoid being misled into making mistakes.
009	DPella employee	Have a front end that represents DPella in color palette	Send this to clients and show uniformity in our products and branding.
010	Forgetful person	Be able to receive a new password	Log in even if I do not remember my password
011	Data Curator	Be able to see my uploaded data sets	Ensure that my uploaded datasets actually were successfully uploaded
012	Data Analyst	Have a list of available datasets	See information about them
013	Data Analyst	Continue with a dataset	Create statistics
014	User	Have a profile page	Edit my account information

1

Figure 2: User Stories

015	Admin	Be able to create accounts	Manage the users effectively
016	Data analyst	Delete one of my available datasets	Do not have to see datasets that aren't relevant for me anymore
017	Health analytics data curator	Delete one of my datasets of patient data	Delete datasets that aren't relevant anymore
018	Data analyst	To see the total budget I have left for a specific dataset	Plan how much budget I want to use
019	Data curator	Trust that the system stops the data analysts when they do not have enough budget	Be confident that the dataset is protected
020	Data analyst	Select which engine I want to run a query with if there is multiple choices	Adjust to my own preferences
021	Data analyst	See which engine I run my query with	Understand what is happening
022	New data analyst	Be able to be informed about what budget is	Make good decisions
023	Busy Analyst	Be able to search on a specific dataset	Find the dataset I want in a fast way
024	Data analyst	See query history of a specific dataset	See what results I have got on the dataset
025	Data analyst	Explore what budget I should use for a query	Optimise the budget I have left
026	Data analyst	Choose which variables I want to use	Create statistics
027	Data analyst	Filter the data	Choose what range I want statistics on
028	Data analyst	Choose what noise mechanism I want to use	Create customised statistics
029	Data analyst	Choose what type of statistics I want to do	Create customised statistics

030	Data analyst	Choose which variables I want to use	Create statistics
031	Data analyst	Be able to get my query result in a histogram	Get more clear results
032	Data curator	Choose the name of the dataset I have uploaded	Keep track of my datasets
032	Data curator	Choose what privacy notation I want to have on a dataset	Create customised datasets
033	Data curator	Set a total budget for the whole dataset	Be sure that the total amount of revealed information is within an acceptable bound
034	Data curator	Choose which variables I want the dataset to have	Customise the dataset
035	Data curator	Assign data analysts to a dataset	Provide DP datasets to the data analysts
036	Data curator	I want my name to be displayed when I log in	experience a personalised interaction
037	Data curator	The program to automatically read variables of the dataset	Upload my dataset quicker and only double check that it seems correct
038	Data analyst	View the variables available in a specific dataset	Select the variables I want to use
039	Data analyst	Be able to perform a query	Obtain specific results
040	Data analyst	Hover over the bar chart	See the exact result
041	Data analyst	Receive an error message when I have not filled in all the required information in the multi-step process	Ensure the creation of accurate statistics
042	Data analyst	Be able to rename the variables	Create customised statistics

043	Data curator	I want to be able to update settings to my dataset	Ensure that my data can be analysed the way I want it
044	User	The stepper to have an improved appearance	Navigate more easily
045	Data curator	Be able to upload the data which is within my uploaded file	So that it can be used for queries.
046	Admin	See a list of all users	Cave a clear overview and manage them effectively
047	Admin	Be able to update user information for a specific user	Manage the user base effectively
048	Admin	Be able to delete users who are no longer relevant	Manage the user base effectively.
049	Admin	Be able to trust that only certain roles can visit specific pages	Can feel confident that all data is protected
050	Data analyst	Be able to do a count in my query	Count the number of rows in the dataset
051	User	The application to be responsive dimensionally	Use the application over different screen sizes seamlessly

User ID.	Task
001	Description: The data analyst should be able to change how much budget they want to use on a specific query. Tasks: 1)Create a slider so the user can adjust the budget 2) Only allow values that are in the correct range 3) Connect the slider to the API endpoint
002	Description:

	The user should be able to read about DP if they are new to the concept or want to refresh their mind. Tasks: 1) Create an informational page explaining differential privacy. 2) Create tooltips that can help uninformed users.
003	Description: The analyst should be able to see the results of the query. Tasks: 1) Implement a page where you can see the result of the query.
004	Description: The data curator should be able to upload their datasets and assign them to data analysts. Tasks: 1)Create button "upload new dataset" 2) Connect to the API end point 3)Only allow certain types of files
005	Description: Tasks: 1) Create a login page 2) Route the user to different pages depending on which role they have.
006	Create a button for log out
007	Create an Escape Hatch in the form of the company logo in the top left corner.
008	Define a colour palette that avoids those that colour blind people have trouble seeing, see here. ←
009	Apply the DPella colour palette to the front-end
010	Implement a password reset feature
011	Display a list of the uploaded datasets with relevant metadata.

012	Description: Show a list of assigned datasets to the analyst Tasks: 1) Make a card which represent the dataset 2) Make a list of the cards 3) Connect to the endpoint
013	Task: Make an arrow on the side of the card that routes to the query page, when clicked on it.
014	Implement a profile page where the user can view and edit their information
015	Create a form for inputting required information when creating a new account.
016	Description: Delete the assigned analyst from the dataset. Tasks: 1) From the list of analysis plans, make a delete button. 2) Ensure that there is a confirmation prompt
017	Description: The data curator should be able to delete a dataset which also means that the assigned analysts are deleted from the dataset. 1) From the list of datasets, make a delete button. 2) Ensure that there is a confirmation prompt
018	Description: Make the total budget left on a dataset always visible for the analysis. Tasks: 1) In the view for available datasets, make the total budget visible. 2) In the view for running a query, make the total budget left visible.
019	Display information to the data analyst when they are not allowed to run a query
020	Allow the data analysts to select from multiple available engines before running a query
021	Display the current engine being used for a query

022	Description: On the page where the data analysts can see and adjust the budget, there should be information about epsilon and what it does. Tasks: 1) Display information in some way (hover, pop-up or something else)
023	Description: Make a search bar on the available datasets page for the data analyst. Tasks: 1) Make a search bar 2) Connect it the available datasets
028	Description: The data analyst should be able to choose which noise mechanism the query should run with. Tasks: Create Laplace and Gauss buttons on the second query page.
029	Description: The data analyst should be able to choose which statistics they want to do, for example sum and count Tasks: Add count, mean, minimum and maximum buttons on the second query page.
034	The data curator should be able to choose which variable the dataset should have Tasks: Create a form where the data curator can set variables and its boundaries.
036	Description: When the user logs in they should see their name

	Tasks: 1) Connect the server to the log in page 2) Make a way to handle user information
037	Description: Read the first row in the file and represent all the variables in a list at the second upload page. Tasks: 1) Implement a file parser 2) Create objects from the variables and list them accordingly in the list
043	Description: It should be able to update the dataset in every possible way which are supported in the API on the same page to a given dataset. Tasks: 1) Connect the dataset name with its unique dataset-id 2) Read through a dataset in order to upload data to it 3) Change name, owner and total budget of the dataset
045	Tasks: 1) Create a file reader which reads all data except from the row with variables 2) Send a request to server with the uploaded data

Appendix B

Name: Elisabet Lobo-Vesga
Date: 08 / 04 - 2024

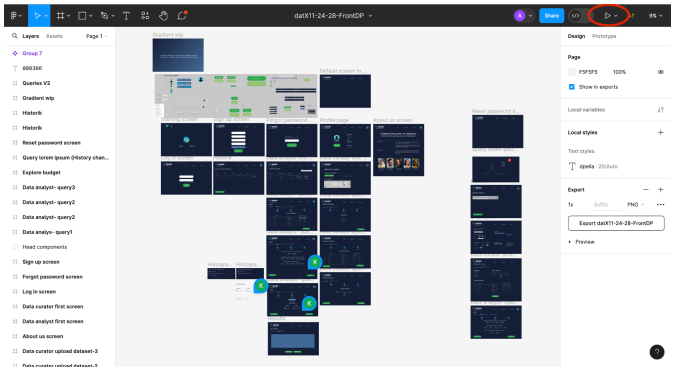
Usability Testing for FrontDP

Welcome to our usability testing session for our interface for DPella! We deeply appreciate your participation and valuable insights as we refine and enhance the functionality of our website.

Your feedback is very appreciated in shaping the app's development. Thank you for being part of this testing session!

Before testing, we need to open Figma which is a designing tool we have used to create a prototype: Click here.

It will look like this, please click on the play button highlighted in the red circle:



A presentation of our prototype will then begin, please proceed to the next page and follow the tasks.

We value any kind of input!

Figure 3: Usability Testing Results

Name: Elisabet Lobo-Vesga
Date: 08 / 04 - 2024

Task 1: Log in to your profile and find information about DPella.

Task 2: Go to your profile page, what is your email address? Log out afterwards.

Task 3: Go back to the Starting page. Sign up as a Data Curator account, what datasets have you uploaded? (Let's pretend that this is not a new curator account)

Task 4: In your Data Curator page, please go through the process of uploading a new dataset. Can you assign multiple data analysts to it?

Task 5: Go back to the Starting page, Log in to your Data Analyst account (Task 1).

Go into the first Dataset:

- How much is left of your budget?
- Select the single-variables "mean", "minimum" and "maximum". After that, explore your budget and create your statistics.

Feel free to explore outside the Tasks provided, please provide your feedback here:

Feedback:

Overall, I believe the prototype is on the right track; depending on how much time the students have to work on this, I could provide more detailed feedback on the overall aesthetics and formatting. This time, I tried to keep my comments more focused on the functionality and clarity of the site and instructions.

When login in, on the available datasets screen:

The test datasets should have different names, test1 and test2 for instance.

When presenting the privacy parameters, perhaps the explanation of the format should come as a subtitle, something like:

epsilon allocated (remaining)	delta allocated (remaining)
...	...

Name: Elisabet Lobo-Vesga
Date: 08 / 04 - 2024

Under the privacy notion column, the options can be simply Pure DP and Approximate DP ; you can drop the "budget" part.

On Task 4:

It might be my inexperience with Figma, but on Task 4, I couldn't assign multiple analysts to my dataset; whenever I clicked on the + button, it highlighted all the other clickable elements of the page.

Similar to my first comment, can we make all the test column names be unique instead of just test ?

On Task 5:

There is a disconnection between the column names (here "variables") presented in the "Confirm variables" stage and the next step. In the "Statistics measure" page, the column "age" comes out of the blue.

In the "Statistics measure" page, when clicking "age", the user is prompted with two options; right now, both can be selected simultaneously. Don't we want them to be mutually exclusive? If this is not the case, we should use checkboxes instead of radio buttons.

In the window query/set budget/explore budget , there is a typo: "confidens" should be "confidence".

There are some inconsistencies in the presentation of the "Confirm variables", "Statistics measure", and "Set budget" pages. The first and last contain their corresponding titles, while the middle ones do not.

<u>General observations:</u>

It would be best to use a gender-neutral name for the test user. In this case, a good option would be Alex or Andrea instead of Angela.

Feedback on WebDP's front-end user-testing 1

The alignment between the text and buttons is off in most cases.

The size of some buttons is not large enough to fit all the information, e,g., when choosing statistics, the ones for "minimum", "maximum", and "histogram" can't fit the check mark once selected.

In query/statistics measure , the options related to histogram bins should only be prompted if the user has selected to perform a histogram.

Appendix C

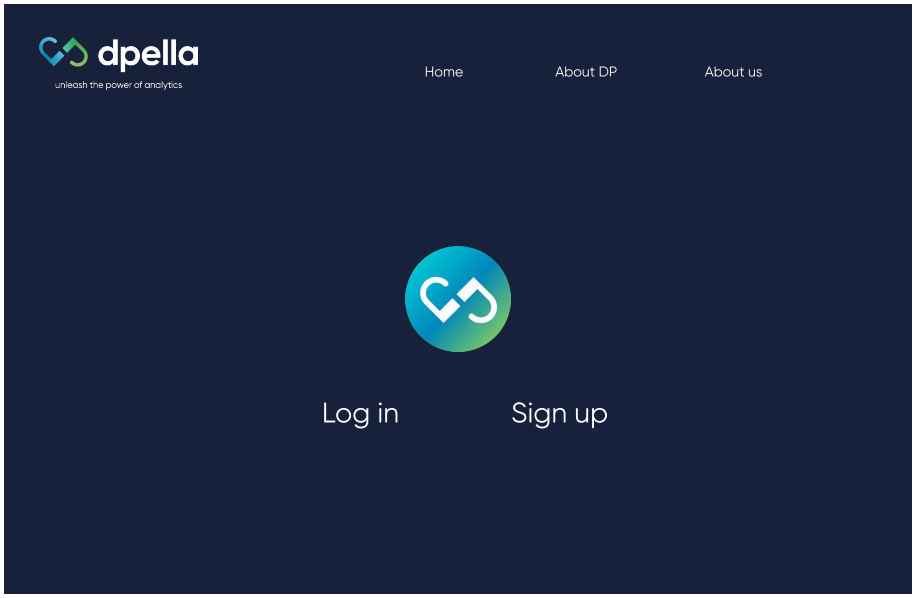


Figure 4: Starting Screen From Figma Prototype

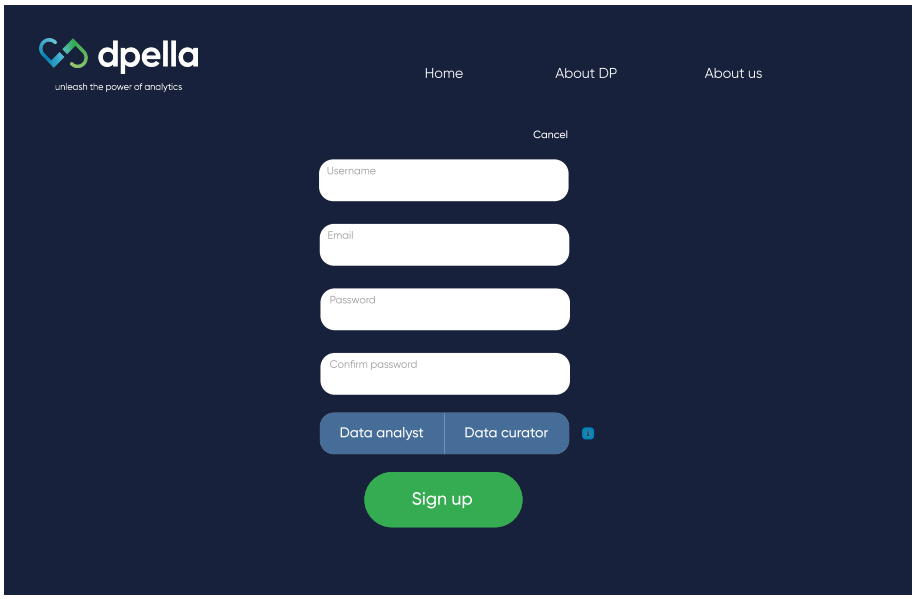


Figure 5: Sign-up Screen

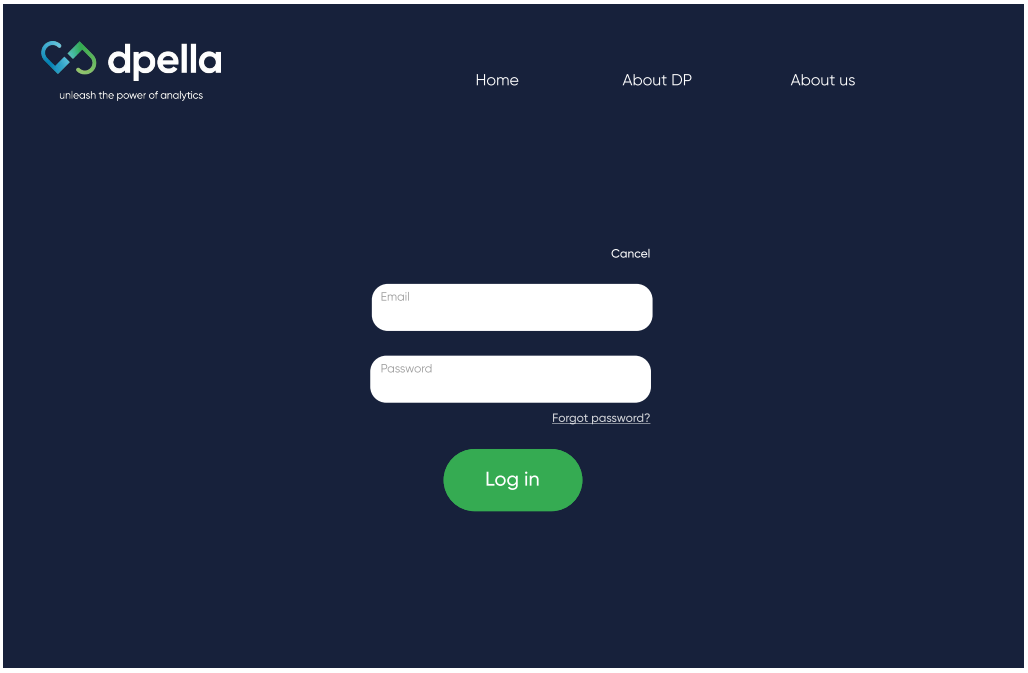


Figure 6: Log-in Screen

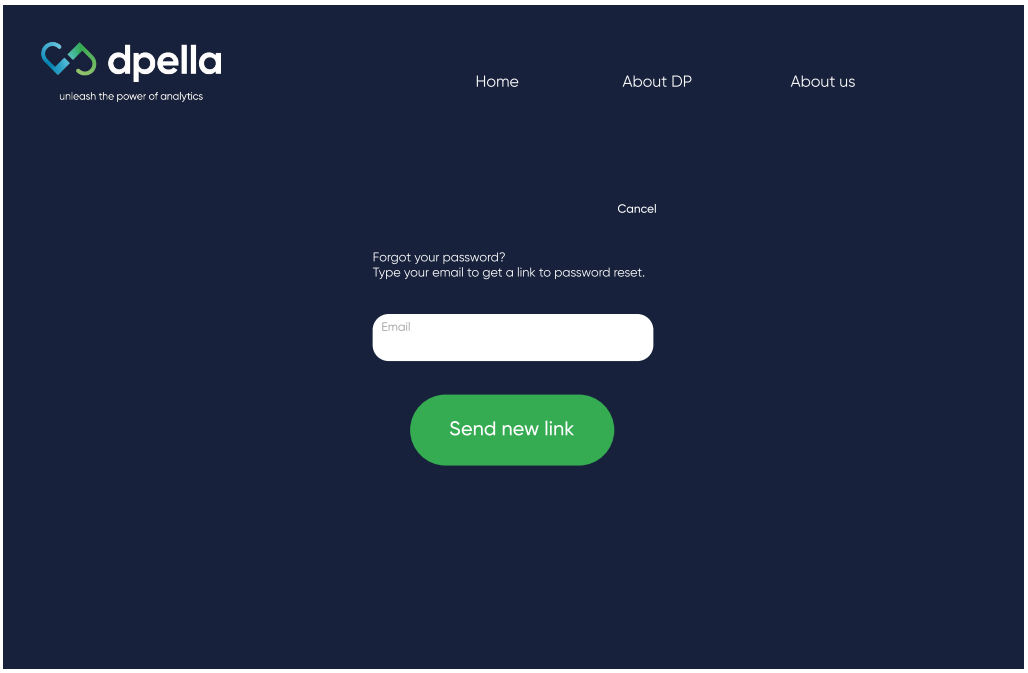


Figure 7: Forgot Password Screen

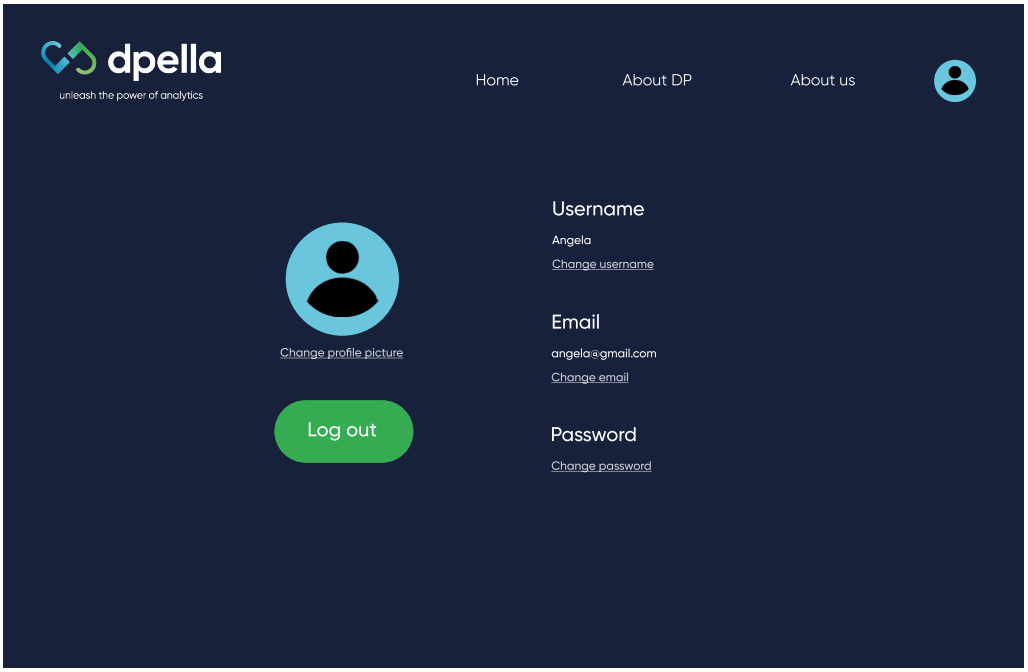


Figure 8: Profile Page

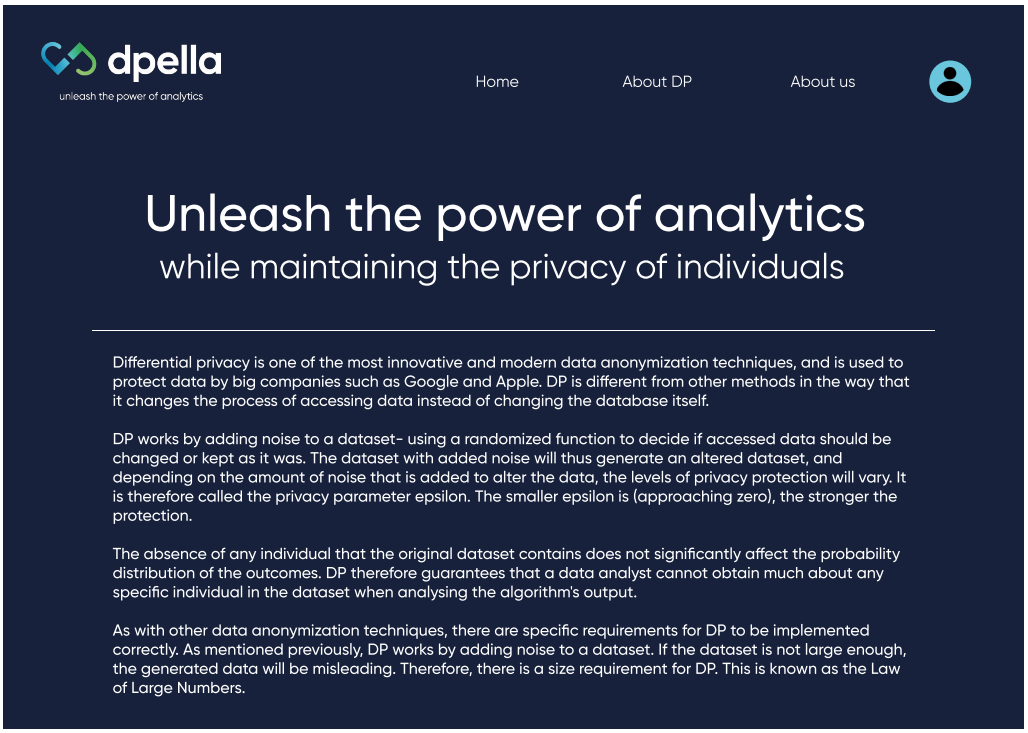


Figure 9: About DP Screen

Figure 10: About Us Screen

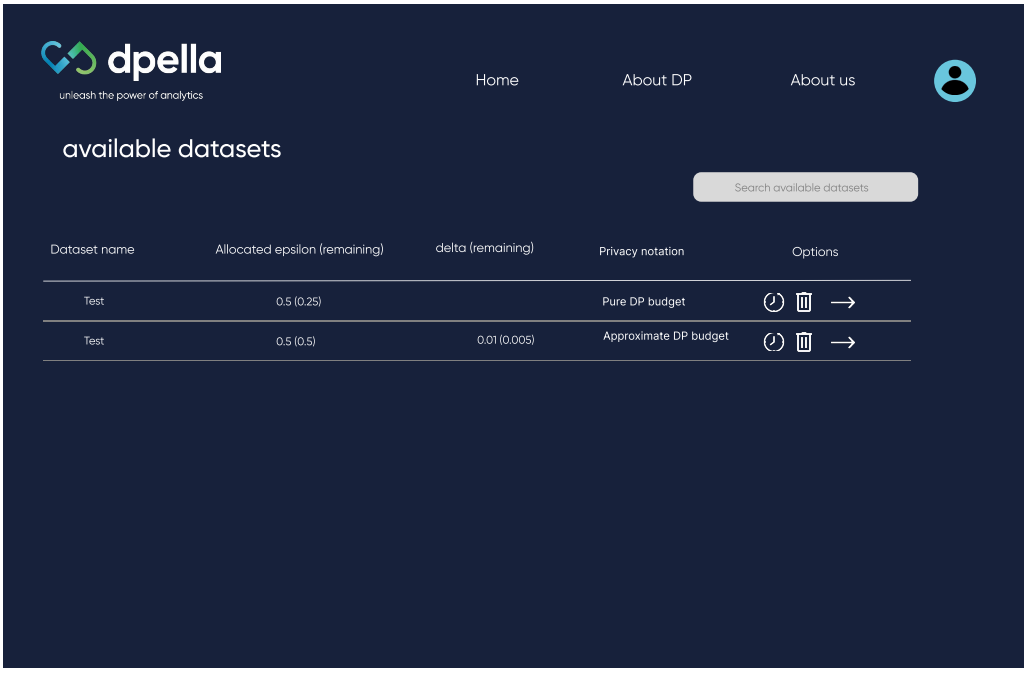


Figure 11: Data Analyst Screen

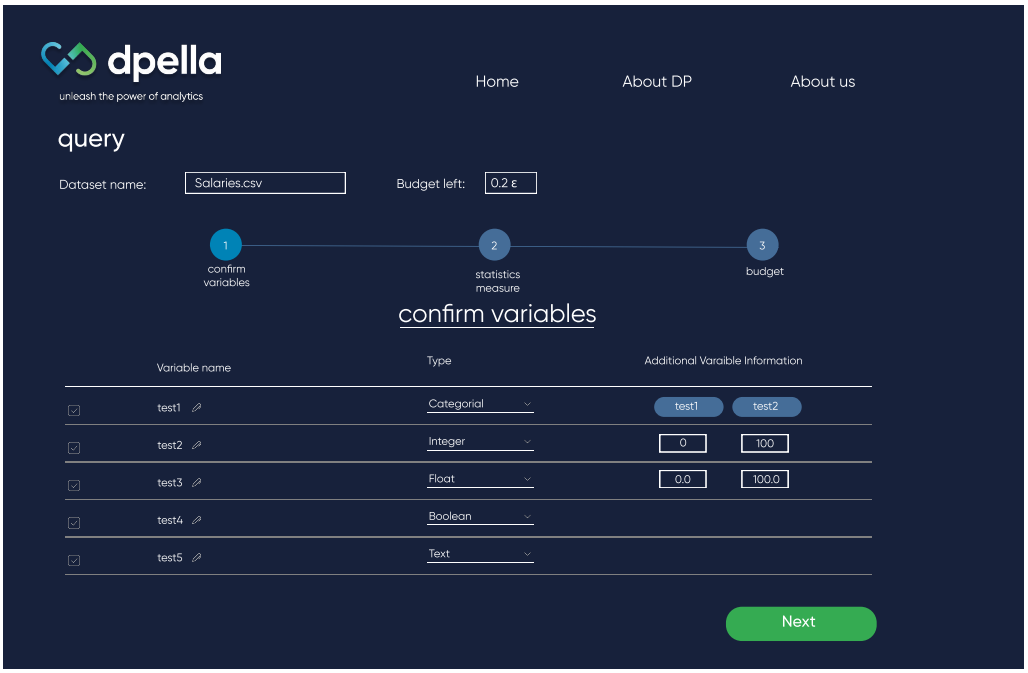


Figure 12: Log-in Screen

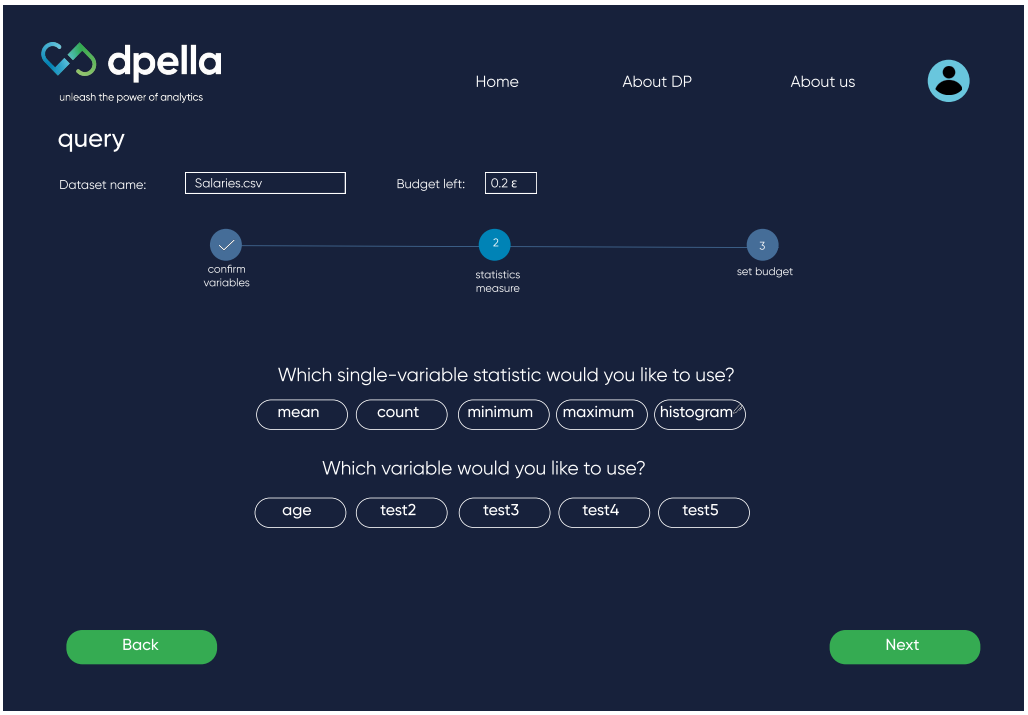


Figure 13: Data Analyst - Query 1

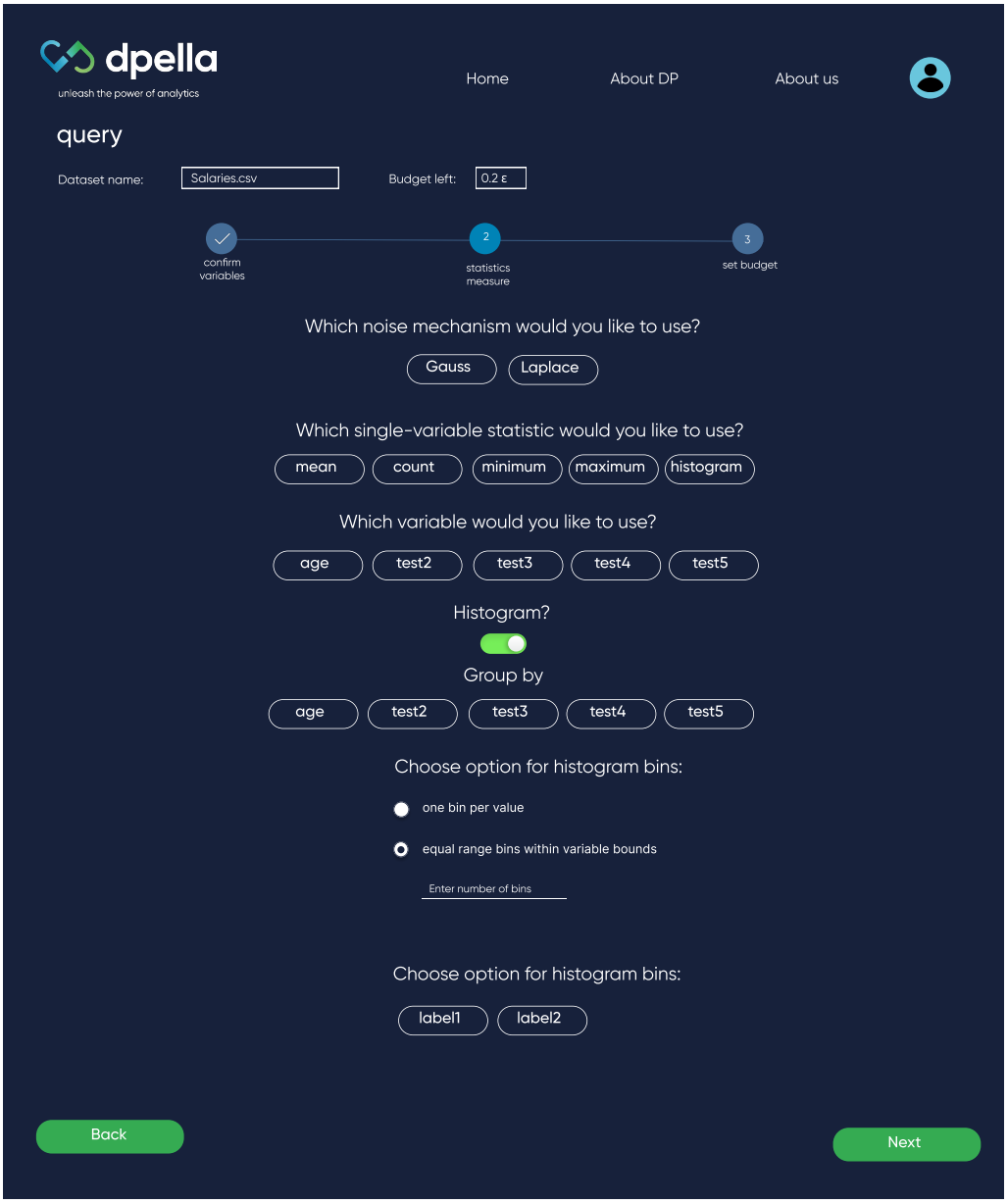


Figure 14: Data Analyst – Query 2

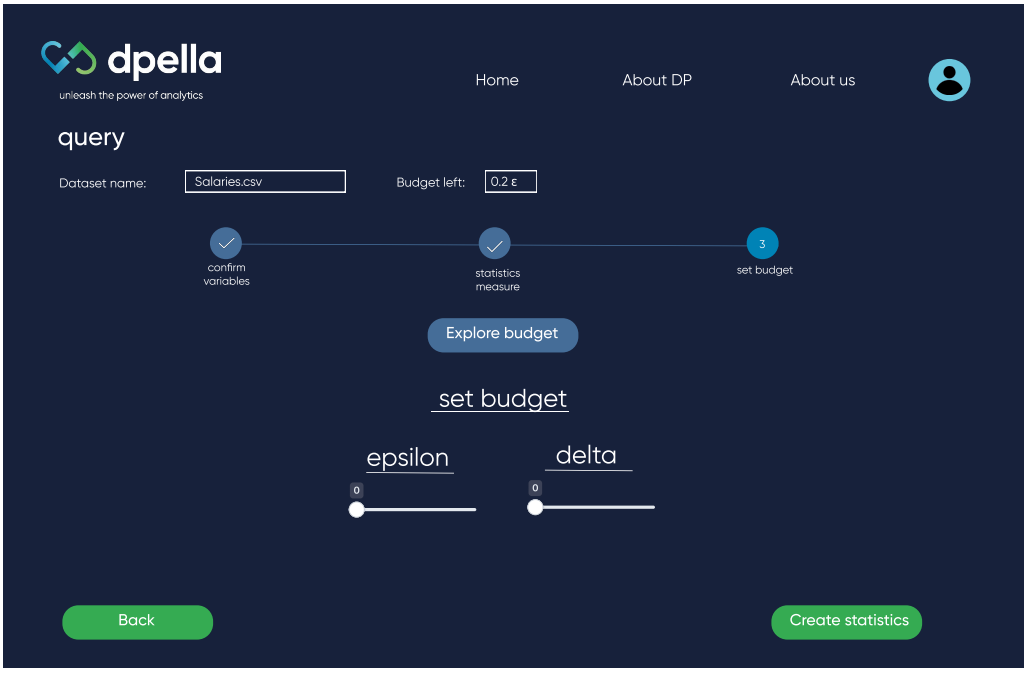


Figure 15: Data Analyst – Query 3

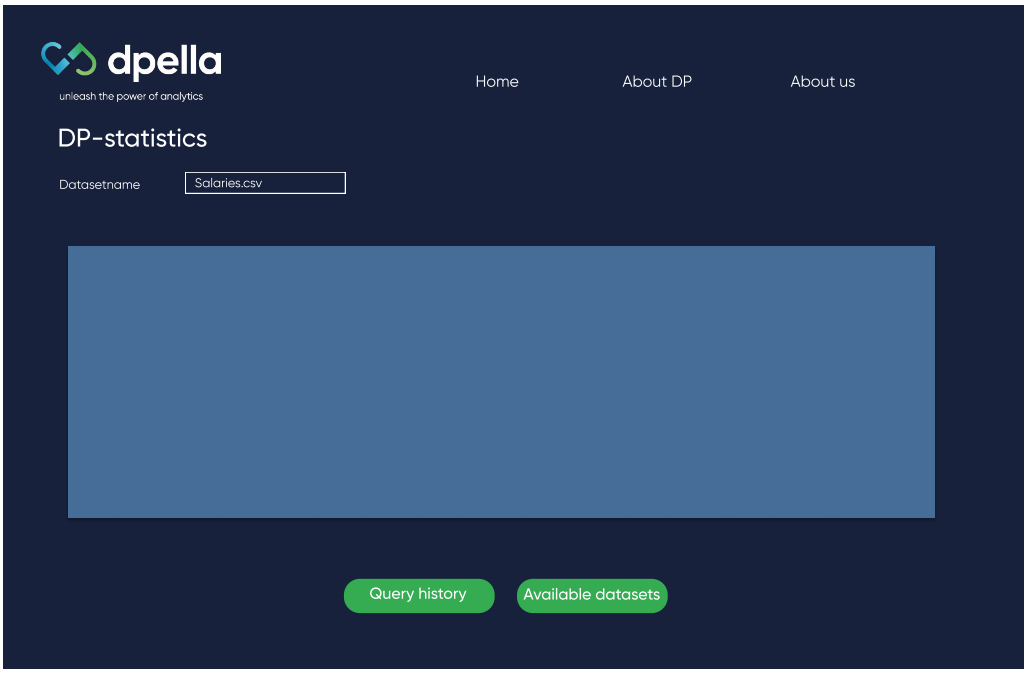


Figure 16: Data Analyst – Results

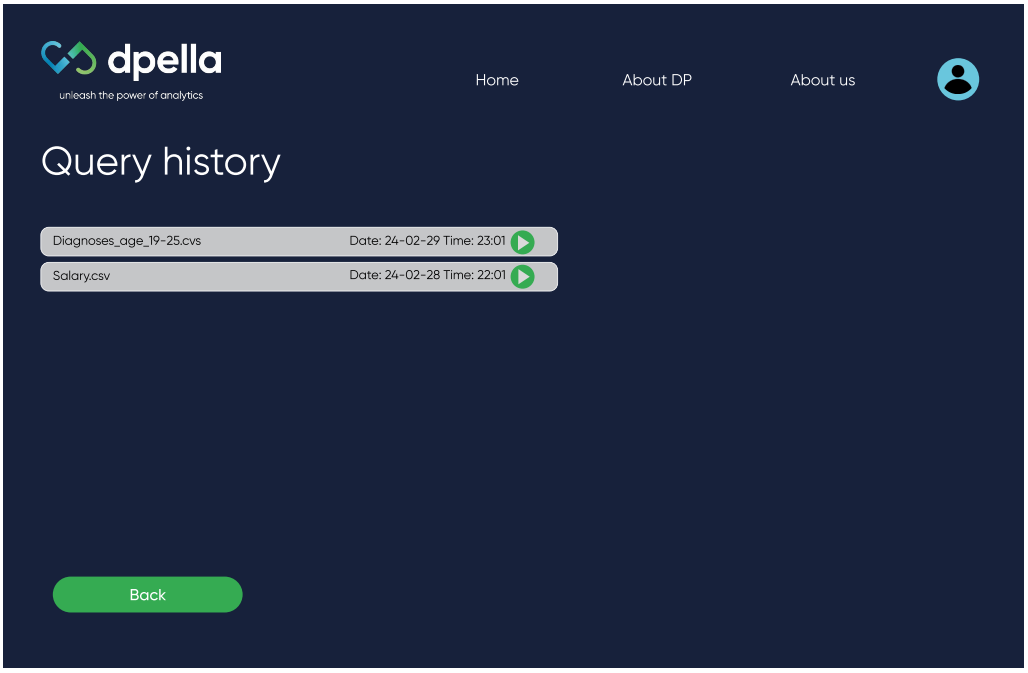


Figure 17: Query History

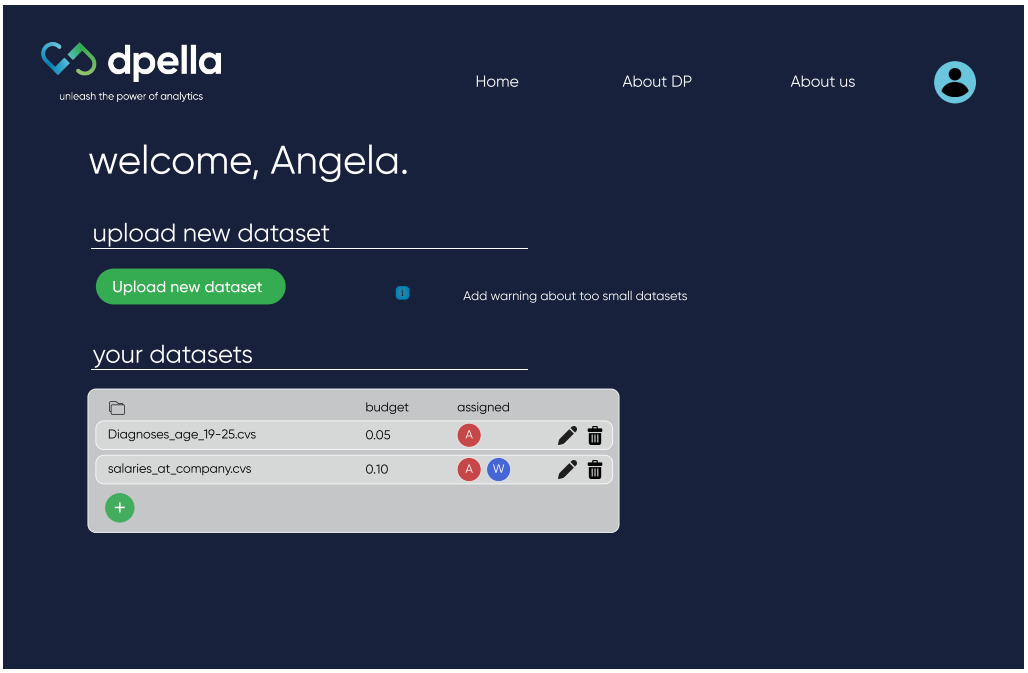


Figure 18: Data Curator Screen

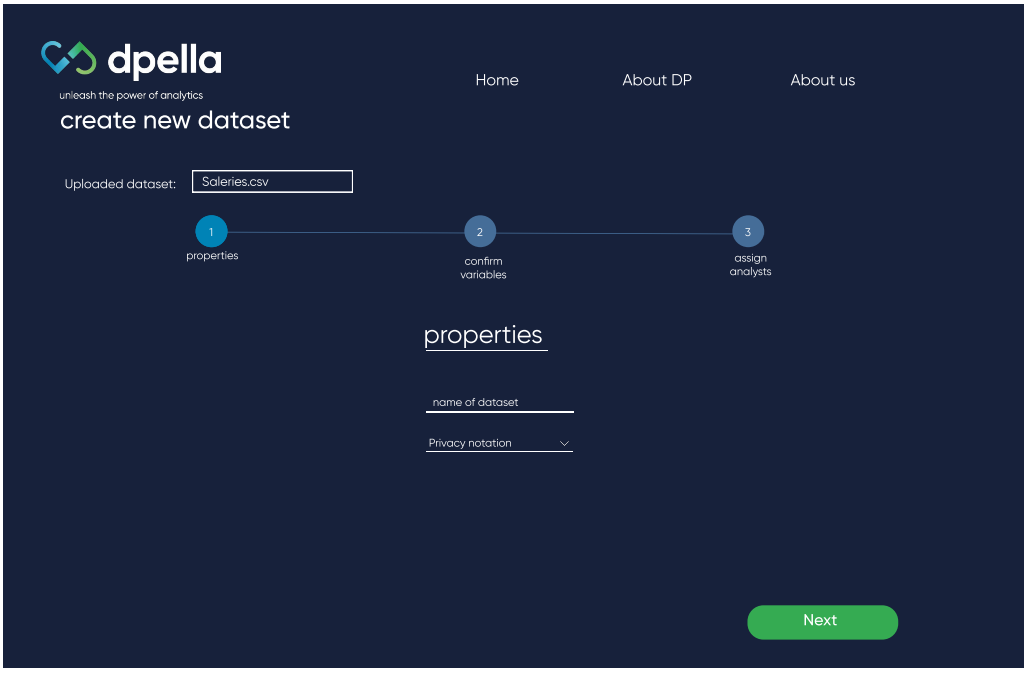


Figure 19: Upload Dataset – Part 1

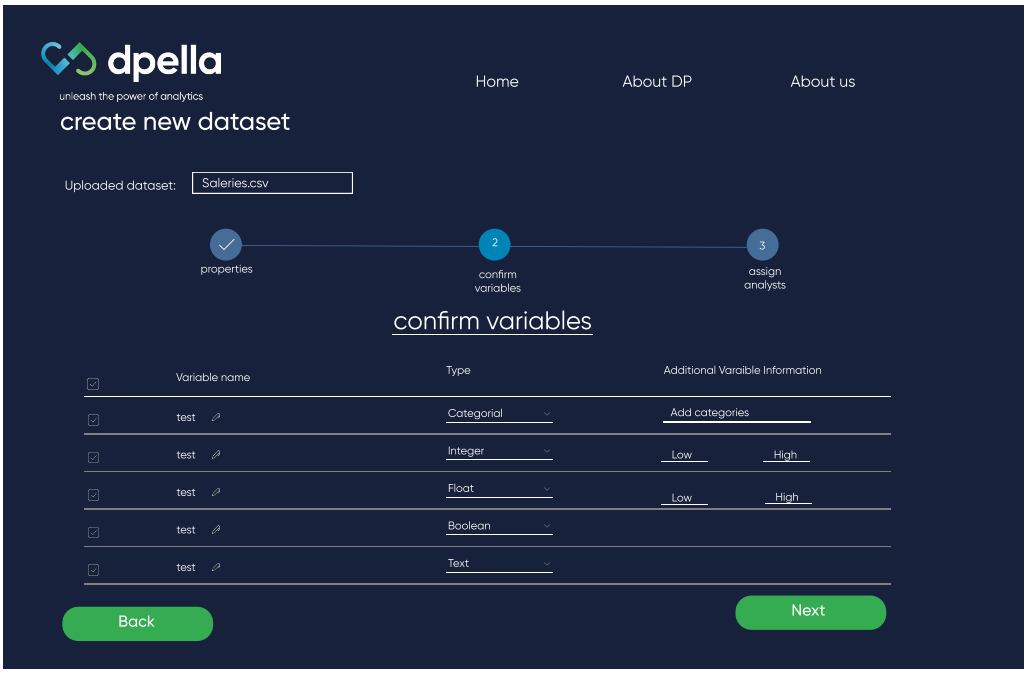


Figure 20: Upload Dataset – Part 2

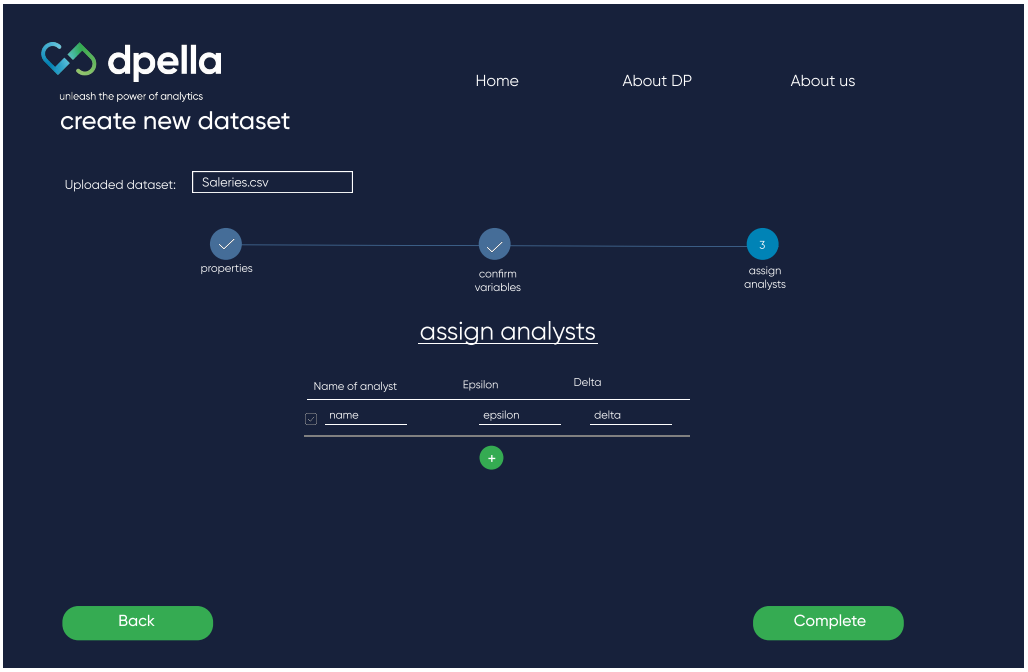


Figure 21: Upload Dataset – Part 3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden

www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY