VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Anouar Belila

REPORTING WEBSITE FOR LOCALTAPIOLA'S DIGITAL SERVICE

Integrating Qlik Sense into a Web Application

Technology and Communication
2024

ACKNOWLEDGEMENT

After a rigorous academic journey spanning five years at VAMK, it is with great pleasure that I embark on the task that would the culminate of my efforts within this thesis. The ever-evolving landscape of information technology presents a range of opportunities and profound potential within today's market, thereby underscoring my choice of this field. My deepest gratitude is extended to my dedicated supervisor, Mr. Anders Wallin of Fellowmind, whose unwavering trust in my abilities has been instrumental in my growth as a developer within the company and, by extension, my involvement in this project.

Furthermore, I wish to express my heartfelt appreciation to the esteemed educators at VAMK, who have played a pivotal role in equipping me with the indispensable knowledge and skills necessary to contribute effectively to a project of this magnitude. Their guidance and commitment to excellence have been indispensable in shaping my academic and professional journey. It is with profound gratitude and a profound sense of accomplishment that I acknowledge their vital contributions to my academic development.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology

ABSTRACT

Author	Anouar Belila
Title	Reporting Website for a Localtapiola's Digital Service Integrating Qlik sense into web application
Year	2023
Language	English
Pages	43 + 3 Appendices
Name of Supervisor	Ghodrat Moghadampour

This project focused on the development of a reporting service for LocalTapiola, an insurance company in Finland, by Fellowmind. The aim of the project was to create a web page using React as a main platform for the whole implementation. The thesis author's role in the project was to act as a main frontend developer, while coordinating with other team members, including backend developers, and a DevOps engineer. The reporting service was later embedded with parent digital service called YRD.

This recent innovative service has significantly enhanced the company's offerings, enabling customers to and analyze interact with visual representations of their expenditures. The advancement of this project, despite encountering substantial challenges such as resource constraints and specification delays, was adeptly managed and overcome. The project's success is attributed to the strategic integration of diverse technologies, including the React library and the Qlik Sense platform, which played a pivotal role in its development.

In conclusion, the topic underscored one of the distinctive methods for embedding data analytics within a web application, seamlessly integrated into a parent portal.

Keywords frontend, data analysis, reporting service, development

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF CODE SNIPPETS

APPENDICES

APPENDIX 1. Information of secondary libraries used in the project.

APPENDIX 2. Long code snippets and their explanations.

1 INTRODUCTION

1.1 Background of the Topic

Fellowmind is a leading Microsoft partner that helps accelerate the digital readiness of customers in various industries. they do this by using Microsoft Cloud solutions, encouraging agile development, implementing integrated platforms, and assisting end-users to learn and adopt.

LocalTapiola, a renowned insurance provider in Finland, offers a diverse range of insurance products to various businesses and individuals nationwide. Customers frequently have to reach out to LocalTapiola's customer service to inquire about their expenses, product details, or to review their insurance purchase history. Recognizing that this process is both time-intensive and resource-demanding for both the customer service team and the clients, LocalTapiola has opted to partner with Fellowmind.

This collaboration aims to create a dedicated reporting service named "YRD reporting service", enhancing the customer experience by streamlining access to vital information. The reporting service, was designed to provide customers with data related to damages, costs, and compensations paid by the insurance company. Through filters, visualizations, graphs, tables, and key performance indicators, the service helps to simplify complex data and provide customers with a clear understanding of their insurance coverage. By offering this reporting service, LocalTapiola is able to provide its customers with a valuable tool to help them make informed decisions about their insurance needs.

1.2 Motivations

There were multiple motivations to pursue this thesis topic for several reasons. First and foremost, this project marked the initial chance to commence from the

beginning at Fellowmind. As a developer, the enthusiasm for tackling fresh challenges and acquiring new skills is constant, and this project offered an excellent chance to accomplish exactly that.

Furthermore, React development is a promising passion, and it can be considered highly rewarding to work on projects that involve this platform.

Finally, Enjoying being part of this project because having the role as the main React Qlik mashup developer was unique. Integrating a custom QlikJS library into React was a complex task that had not been previously attempted at Fellowmind. However, through the efforts, the integration was successfully accomplished, which was a significant accomplishment for the company as a whole. These factors combined have been a motivation to delve deeper into this project through a thesis research.

2 RELEVANT TECHNOLGIES

The YRD reporting service utilizes a diverse range of technologies, platforms, APIs, and sources to achieve its objectives.

React library is a popular JavaScript library for building user interfaces, primarily for web applications. Developed and maintained by Facebook, it enables developers to create large web applications that can update and render efficiently with changing data, without reloading the page. React's key feature is the concept of components, which are reusable, self-contained modules representing a part of the user interface. Each component has its own state and lifecycle, and React's declarative nature makes it easy to predict how the UI will look and behave in response to data changes. React also optimizes rendering performance, ensuring smooth and interactive experiences. This library has gained a vast ecosystem, including tools and extensions like React Native for mobile app development, making it an integral part of modern web development [2]. The library is favoured for its efficient, declarative, and component-based approach, offering greater flexibility and performance compared to other frameworks, especially in dynamic user interfaces.

Webpack an open-source JavaScript module bundler. It's primarily used for bundling JavaScript files for usage in a browser, yet it's capable of transforming, bundling, or packaging just about any resource or asset. Webpack processes application assets like JavaScript, HTML, and CSS. The bundler executes from a configuration file where the behaviour of assets and directories is defined. It enables developers to work with modules in their project and then bundles these modules into a small number of bundled assets, often just one, to be loaded by the browser. This significantly improves performance and load times. Webpack also offers features like code splitting, which allows loading parts of the application on demand. It has become a key tool in modern web development, especially in complex front-end applications [3].

The main advantage of Webpack is when used for its powerful module bundling, enabling efficient management of assets and dependencies. It excels in optimizing load times and offers extensive customization, surpassing other tools in flexibility.

Contentful is cloud-based, headless Content Management System (CMS) that offers a platform for storing, managing, and delivering content to various types of applications and devices. It operates on an API-first approach, meaning that content is accessible through APIs for display on any platform, such as websites, mobile apps, or interactive displays. This system separates the content from its presentation, allowing for greater flexibility in how and where the content is used. Contentful supports a content infrastructure model, providing tools to define the structure of the content, manage it in a centralized location, and deliver it via APIs. It enables editors to create and organize content in an intuitive interface, while developers can access this content through Contentful's Content Delivery APIs, using the programming languages and frameworks of their choice [4]. Hence, Contentful is favoured for its flexible, API-driven content management, allowing seamless integration with various platforms and devices. Its main advantage over traditional CMS frameworks is the decoupling of content from presentation, facilitating omnichannel content delivery and enhancing the agility and scalability of digital content strategies.

Qlik Sense platform is self-service data visualization and business intelligence tool that allows users to analyse and visualize data to reveal insights. It provides an interactive, user-friendly interface where users can drag and drop data to create customizable reports and dashboards. Qlik Sense's unique associative engine indexes and connects relationships between data points across multiple sources, allowing users to explore data deeply without predefined queries. This platform supports a wide range of data sources, including big data streams, cloud sources, and on-premise data. It's designed to cater to a diverse user base, from individual users to large enterprises, enabling them to make data-driven decisions. Qlik Sense also offers collaborative features for sharing insights and working together on data

analysis, along with robust data governance and security measures [5]. Qlik Sense is generally favoured for its powerful associative data modelling, which enables comprehensive data exploration without predefined queries. Its main advantage lies in its user-friendly interface and self-service capabilities, allowing users with varying technical expertise to create dynamic visualizations and gain deep insights from complex data sets.

Azure DevOps is a suite of development tools provided by Microsoft, designed to support a range of software development needs. It encompasses a broad set of services, including Azure Boards for project planning and tracking, Azure Repos for version control, Azure Pipelines for continuous integration and deployment, Azure Test Plans for manual and automated testing, and Azure Artifacts for package management. This platform facilitates collaboration among software development teams, enabling them to plan work, collaborate on code development, build, and deploy applications. Azure DevOps supports a variety of programming languages and integrates with numerous tools and services, both from Microsoft and third-party vendors. It's designed to be flexible and scalable, catering to the needs of teams of all sizes and adaptable to any type of project, whether it's a small web application or a large, complex system. Azure DevOps can be used as a cloud service (Azure DevOps Services) or as an on-premises solution (Azure DevOps Server) [1]. This DevOps environment is favoured for its comprehensive suite of development tools that support the entire software development lifecycle. Its main advantage lies in seamless integration with Microsoft's ecosystem and services, offering robust scalability, security, and a unified experience for planning, development, testing, and deployment processes.

Module federations is a feature of the webpack module bundler that allows JavaScript applications to dynamically load code or modules from other applications at runtime. This advanced architecture enables different applications, or parts of applications, to share components and functionality without needing to compile

them into a single bundle. It essentially allows for separate webpack builds to function as a cohesive system, each independently deployable but capable of sharing live code, libraries, or components with one another. This approach is particularly useful in microfrontend architectures, where different frontend applications, possibly developed by separate teams or using different technologies, need to work together seamlessly. Module Federation facilitates the creation of scalable and maintainable frontend ecosystems by supporting the integration of disparate codebases in a flexible and efficient manner [6]. This feature is favoured to create scalable, decoupled micro-frontend architectures. Its main advantage is enabling different frontend applications to share code and components in real-time, enhancing flexibility and efficiency in development and deployment processes.

These technologies represent the core components of the frontend infrastructure. However, in the comprehensive full-stack perspective, including Qlik Sense and a range of other resources, several additional technologies were used. The oversight of these technologies was the responsibility of other developers on the team, and due to privacy-related concerns, it was not possible to possess authorization to access the specifics of these additional technologies.

3 APPLICATION DESCRIPTION

This project is a reporting service named “YRD reporting service”, which provides a data visualization page for insurance costs. Users can view an overview of their customers' insurance costs based on the data they are authorized to control, filter, and see. The data is fetched from the Qlik sense application created by Qlik sense developers, and authorization and login services are done in collaboration with the backend endpoints. The web page needs to be converted into a web component, so it can be embedded into the parent application abbreviated as "YRD portal". The reporting service application has multiple objectives and specifications, such as supporting three languages, fetching text from the Contentful API, creating a clean and clear code, providing a suitable development environment, ensuring smooth and fast performance, and adhering to Web Content Accessibility Guidelines. To achieve these objectives, various technologies, platforms, and APIs have been implemented, such as React, Webpack, .NET 6, Con- tentful, Qlik Sense, Azure DevOps, and Azure Virtual Desktop.

3.1 Requirements

Requirements during the whole process have not been defined at once at the beginning of the implementation instead they were given during multiple stages. These are the main prerequisites as discussed next.

Initially, a crucial requirement for this project is the establishment of a coordinated development environment in collaboration with the Localtapiola IT team. This environment will provide all project members with access to necessary Localtapiola IT services such as its Qlik server and Confluence, while developers will have the essential tools to facilitate development and testing.

Moreover, to gain access to services provided by Localtapiola, including the Azure machine, all project members are required to have valid External credentials such as emails and usernames.

Additionally, the protection of customer's data is of the utmost importance to this project. Fellowmind and external developers must maintain the privacy customer's data and refrain from sharing it with any other entity.

A secondary requirement in the project is going through all the LocalTapiola's developer tools which are a set of documents and libraries that help a developer complete all the steps in accordance with all the guidelines provided by Localtapiola.

It is also advantageous for a frontend developer to have a Qlik Sense server installed on their machine. This setup allows them to refine and enhance their skills in Qlik Sense mashup development within a personal environment. Here, concerns regarding data protection are minimized prior to actual implementation in the development environment, facilitating a more secure and efficient learning process.

The requirements have been prioritised based on the following table.

Table 1. Table of requirements.

Requirements	**Priorities**		
	Must have	Should have	Nice to have
Development environment		X	
LocalTapiola's credentials	X		
Data protection	X		
Access to LocalTapiola's developer tools			X
Local Qlik sense server			X

3.2 Use-case Diagram

The use-case diagram provides a detailed representation of the functionalities of a typical user, possessing full access rights, in the LocalTapiola's reporting service webpage. Depending on the user's role, various sensitive case scenarios may restrict certain functionalities. For instance, certain users may have limited access to specific graphs, key performance indicators (KPIs), or, in exceptional cases, an entire page.

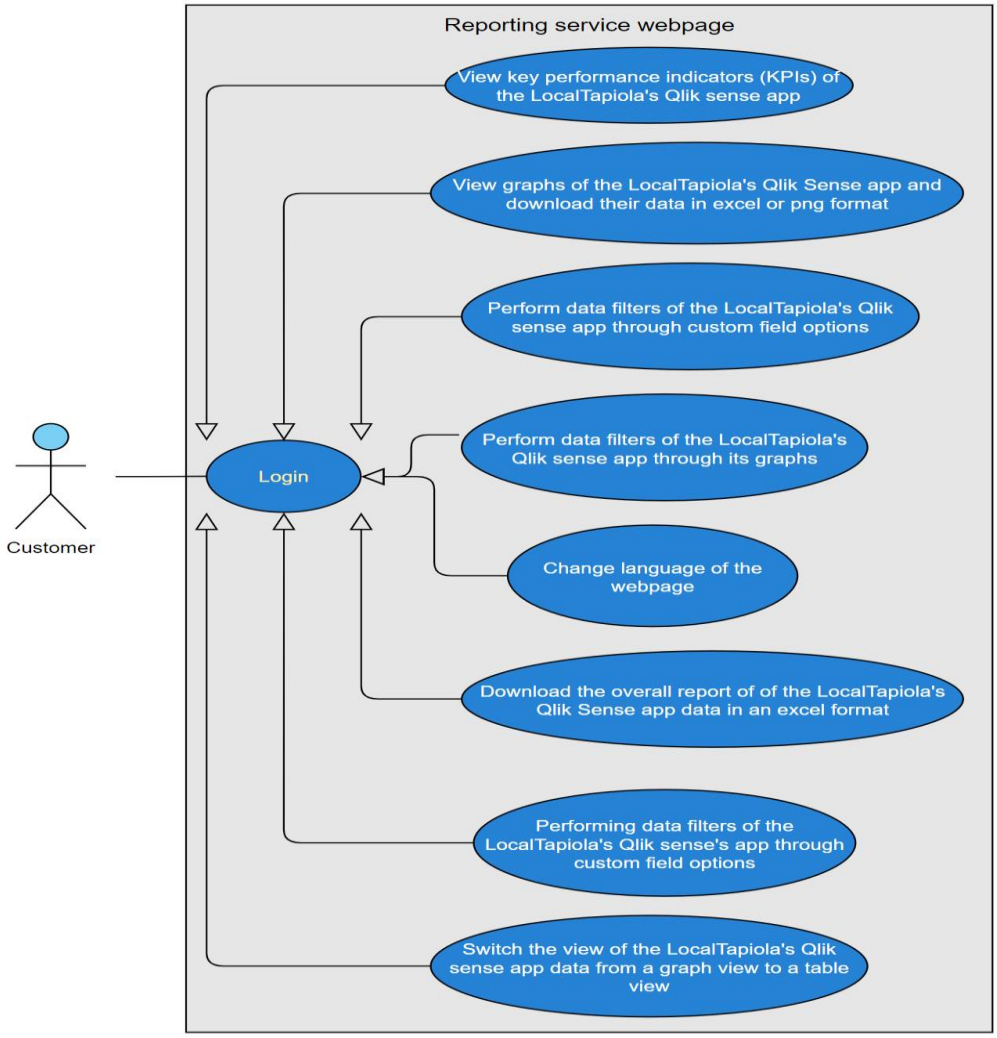

Figure 1. Functionalities provided by LocalTapiola's reporting service.

3.3 Sequence Diagram

The architectural diagram illustrates the process the login then establishing a connection between the Qlik mashup and the Qlik sense server. During this phase, two pivotal resources, namely qlikstyles.css and require, are retrieved, followed by consistent communication facilitated through a WebSocket session, then all data of the LocalTapiola's Qlik sense application are fetched.

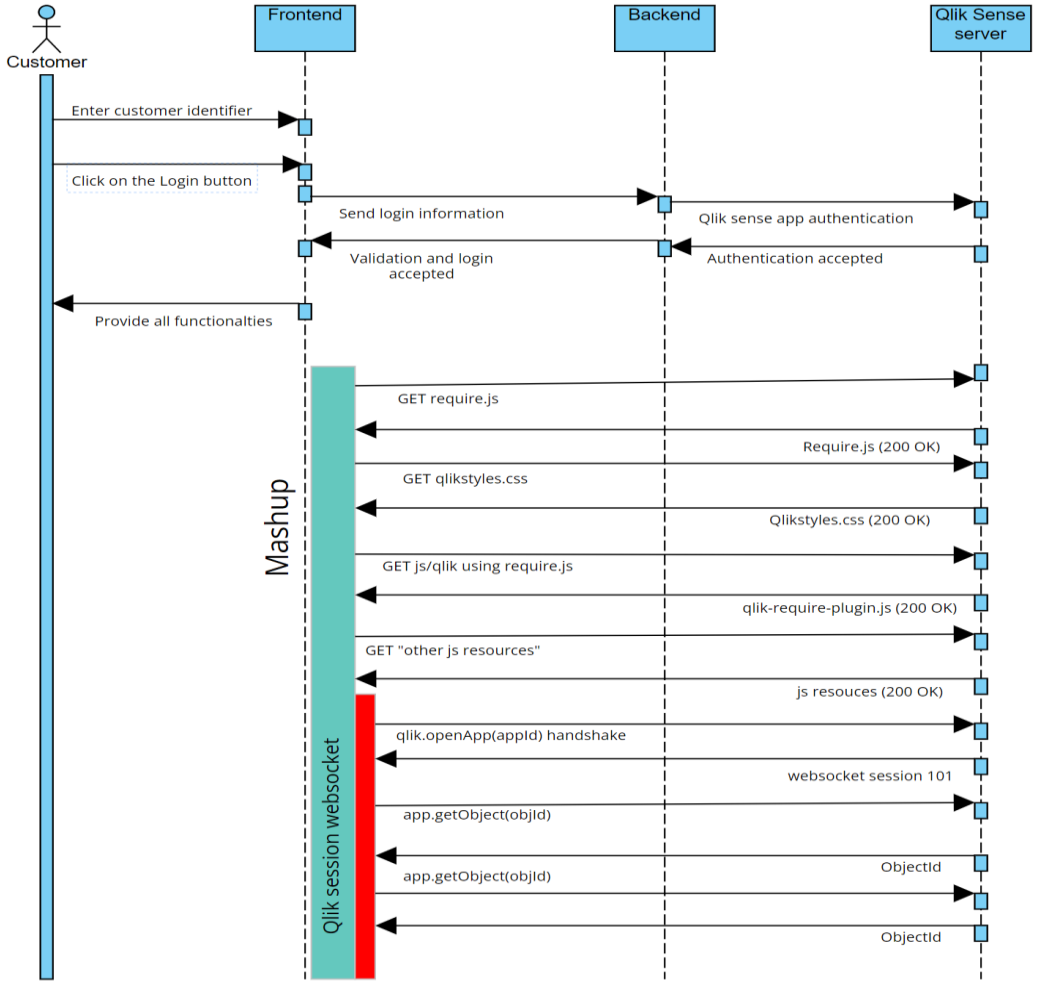

Figure 2. Connection process between frontend and Qlik sense server.

3.4 Component Diagram

The following figure illustrates the component hierarchy within the React application. It depicts a Router containing two sections that share certain components. However, the arrangement and layout of these shared components may vary between the two pages, contingent upon the data passed to the respective components, as depicted in the figure.

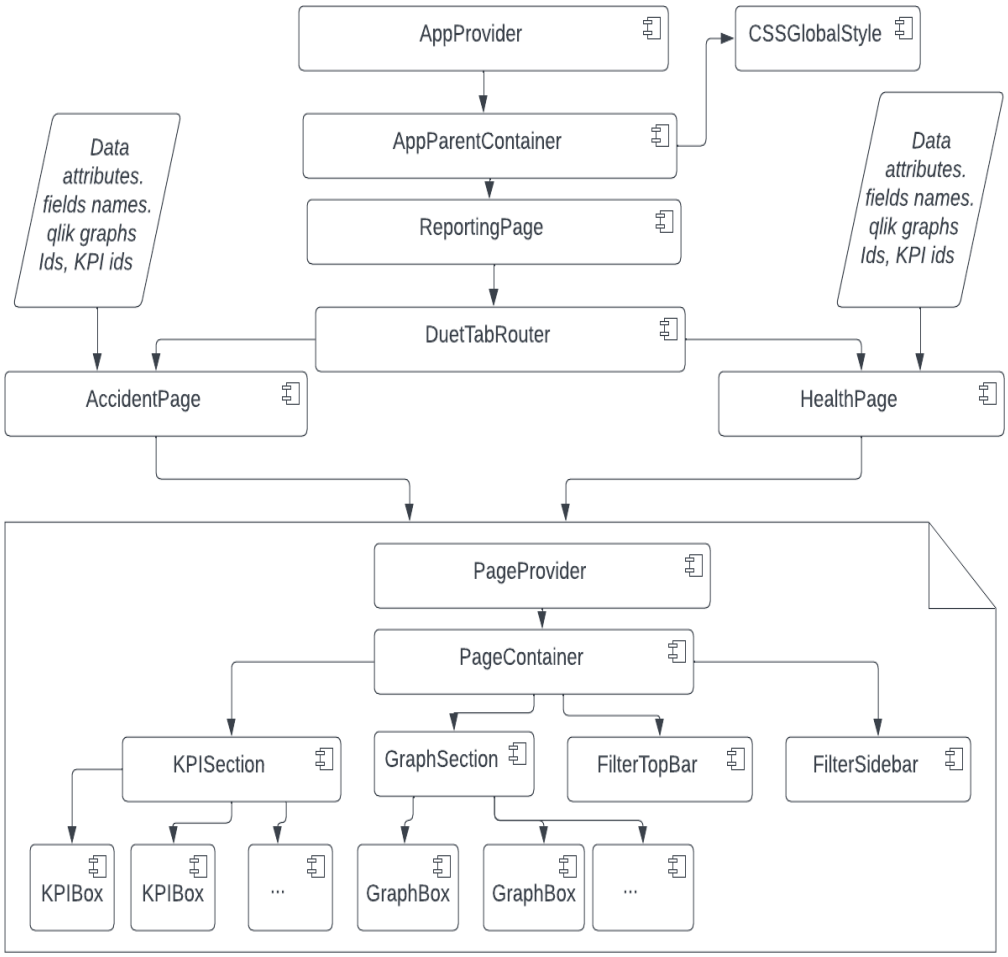

Figure 3. The structure of components in the frontend.

4 GUI DESGIN

4.1 Design of Different Parts of GUI

During the initial stages of development, various design proposals were put forth; however, the ultimate design concept was achieved through a process of coordinated collaboration and unanimous approval among all participating designers. The graphical representations were obtained from the mock-up model accessible within the Figma platform at the customer level. The components of the design as presented next.

The webpage depicted in the below figure primarily features two tabs: one for the Accident page and the other for the health page. Both pages share identical layouts. At the top of each page, there is a span that displays the date of the most recent information update. This is followed by the title and description of the page. Subsequently, there is a multi-selection tab group labelled "Circumstances." Below this, two buttons are situated: one to reveal additional filters and the other to download the overall report.

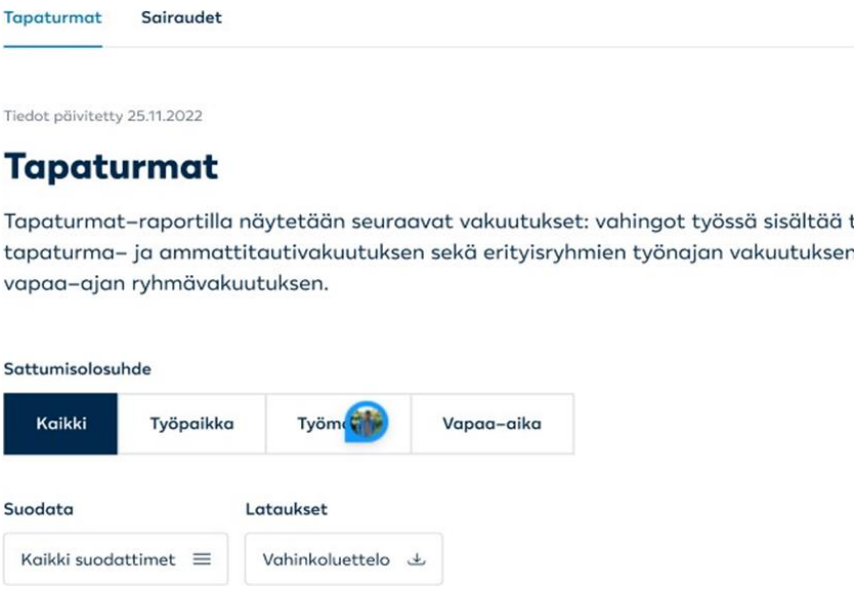

Figure 4. The entrance page of the application

The key performance indicators (KPIs) illustrated in the figure below are situated beneath the two buttons referenced in Figure 4. These indicators comprise several boxes, each containing a title, a value, and a modal for accessing further information about the specific KPI. All data, including the values and titles, are sourced from LocalTapiola's Qlik Sense application.

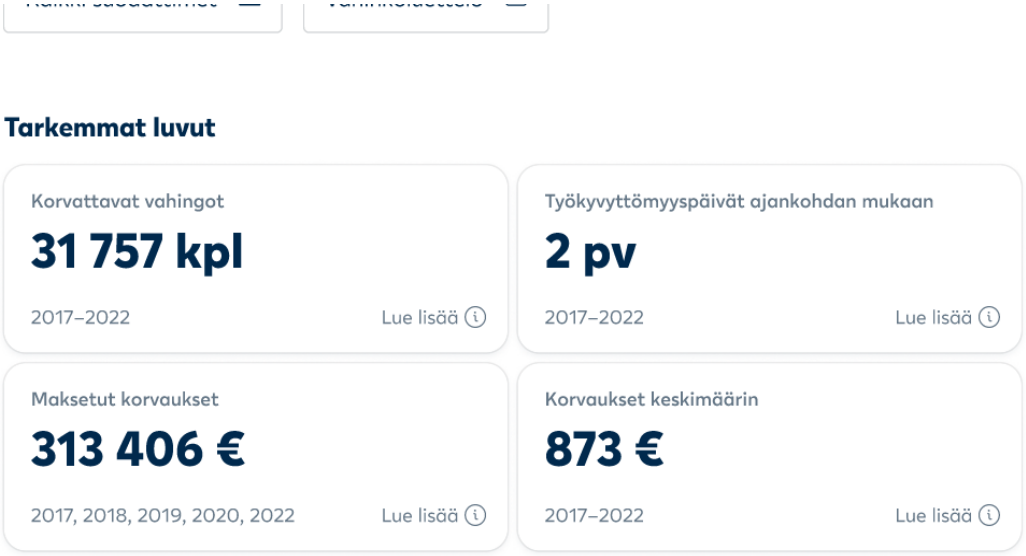

Figure 5. Key performance indicators section

The "Graph Boxes" section, referenced in Figure 5, is located beneath the previously mentioned section. This area predominantly features a variety of graphs. The two graphs displayed below serve as examples of the typical appearance of these graph boxes. Each box includes the title of the graph and the graph itself. At the bottom of each graph, there are two buttons enabling users to download the graph as an image or export the graph's data as an Excel file. Additionally, a toggle is provided to switch the display from a graph to a table view.

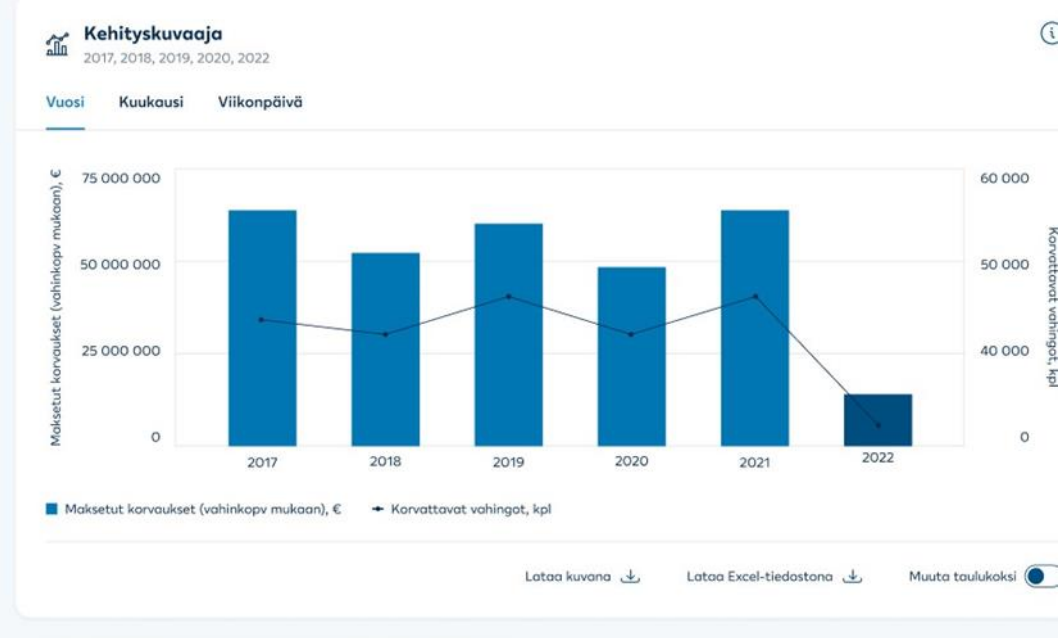

Figure 6. Custom graph box

Figure 7. Custom graph box

Upon selecting the filters button displayed on the entrance page, the filter section, as shown in Figure 8, emerges as a sidebar on the right. This section primarily consists of various collapsible elements, allowing users to apply selections and filters based on the options available within each field.

Figure 8. Filter section

As depicted in Figure 9 below, this modal provides additional information. Within the key performance indicator (KPI) box, as shown in Figure 5, there is a button designed to reveal more details about the KPI. The modal includes a title and description for each KPI, followed by a section at the bottom displaying frequently asked questions.

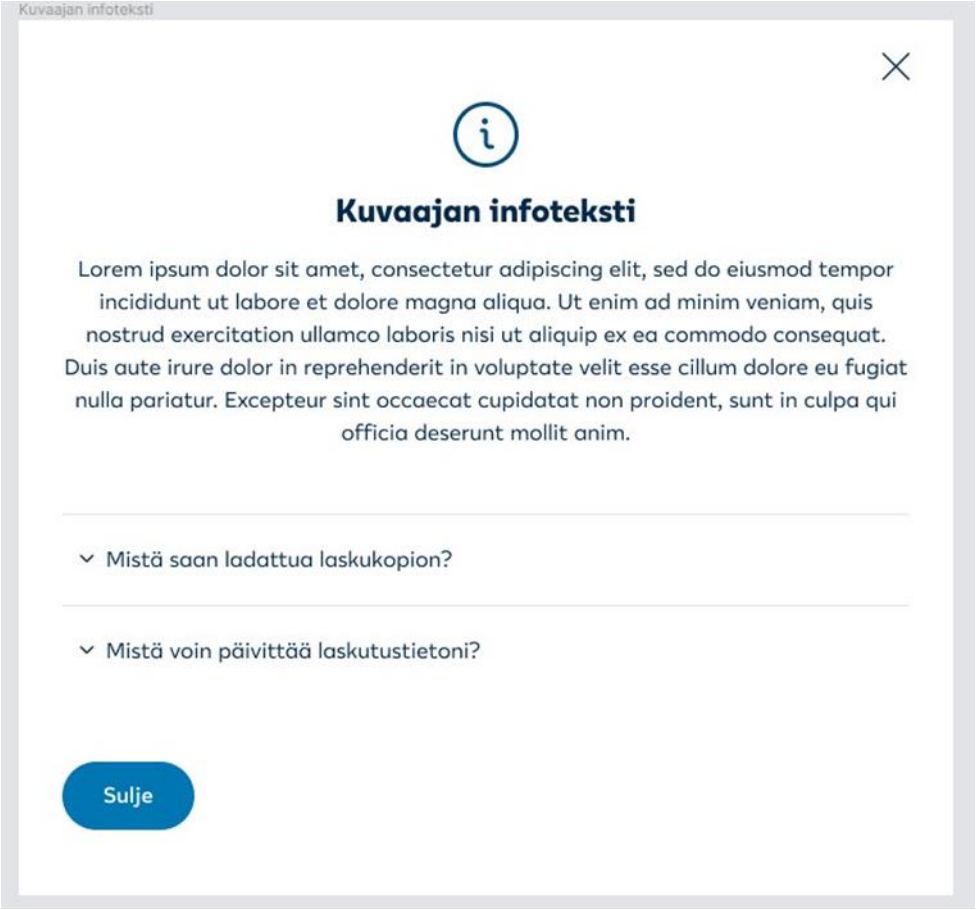

Figure 9. Learn more modal

The entrance page of the Turva version of the reporting service is presented in Figure 10 below. This Turva version is largely similar to the main version, with the primary distinctions being in the themes and colour schemes used. While the main

version of the reporting service employs light blue, the Turva version utilizes a combination of red and dark blue.

Figure 10. Entrance page for YRD reporting service of Turva version

5 IMPLEMENTATION

5.1 General Description of the Implementation

In the project implementation, various steps were undertaken, including Qlik Sense Server installation, custom QlikJS library creation for React, Figma design component implementation, React-Qlik Sense app connection establishment, creation of mashup components with Qlik Sense connections, and the introduction of a mechanism for sensitive data protection. Additionally, a distinct Turva version was implemented with modified colours and themes.

5.2 Implementation of Different Parts

First, installing of Qlik Sense Server on the local machine, which allowed for the uploading of a Localtapiola's Qlik Sense application with prototype data to aid in development.

Second, creating a custom QlikJS library for React, which was adapted from an AngularJS JS module to ensure compatibility with React. The library is built upon a TypeScript file comprising of 1000 lines, featuring interfaces, enums, and types. Due to its size, only the most crucial interface from the file will be showcased.

```
export interface QlikApp {
  addAlternateState(qStateName: string): Promise<any>;
  back(): Promise<any>;
  clearAll(lockedAlso?: boolean, state?: string): Prom-
ise<any>;
  close(): void;
  createCube(
    qHyperCubeDef: HyperCubeDef,
    callback?: (hypercube: HyperCubeResponse) => void
  ): Promise<any>; // TODO: Returns Promise<object Model>
  // TODO: createGenericobject
  createList(
    qListobjectDef: ListobjectDef,
    callback?: (hypercube: any) => void
  ): Promise<any>;
```

```
  destroySession(id: string): Promise<any>;
  destroySessionObject(id: string): Promise<any>;
  doReload(
    qMode?: "0" | "1" | "2",
    qPartial?: boolean,
    qDebug?: boolean
  ): Promise<any>;
  doSave(qFileName?: string): Promise<any>;
  field(field: string, state?: string): QField;
  forward(): Promise<any>;
  getAppLayout(callback: (layout: Layout) => void): Prom-
ise<any>;
  // getAppobjectList(type: 'sheet' | 'masterobject',
callback: (list: ))
  getFullPropertyTree(id: string): Promise<any>;
  getList(type: ListTypes, callback?): Promise<any>;
  getObject(
    elem?: HTMLElement | string,
    id?: string | "CurrentSelections",
    options?: { noInteraction?: boolean; noSelections?: bool-
ean }
  ): Promise<any>;
  getObjectProperties(id: string): Promise<any>;
  getSnapshot(elem?: HTMLElement | string, id?: string): Prom-
ise<any>;
  lockAll(state?: string): Promise<any>;
  removeAlternateState(qStateName: string): Promise<any>;
  serachResults(qTerms: any[], qPage)
  // searchSuggest(qTerms: any[], qOptions: { qSearchFields:
any[] }, ): Promise<any>;
  // selectAssociations
  selectionState(state?: string): any; // QSelectionState;
  unlockAll(state?: string): Promise<any>;
```

Code snippet 1. Interface of Qlik custom library.

The above code snippet contains an interface exclusively displays the most crucial methods within the Qlik custom library. In the actual source code, there likely exists an extensive codebase exceeding 1,100 lines. Consequently, for the sake of report clarity and convenience, not all lines have been presented in the interface.

Third, implementing the primary components of the mock-up design.

Then, establishing connection between the React App and the LocalTapiola's Qlik sense app that is hosted in LocalTapiola's Qlik server. this step was done in coordination with backend developer. The connection was established within the parent component label of React, referred to as the context. This section presents the crucial components of the code responsible for facilitating this connection.

```
    if (app === null) {
      (async () => {
        const { isSecure, host, port, prefix } = config;
        const protocol = isSecure ? "https://" : "http://";
        const portStr = port ? `:${port}` : "";
        const baseUrl = `${protocol}${host}${portStr}${pre-
fix}resources`;
        const requireConfig = {
          baseUrl,
          paths: {
            qlik: `${protocol}${host}${portStr}'/qlik/re-
sources`,
          },
        };
        //@ts-ignore
        if (booleanConvert(process.env.RE-
ACT_APP_FETCH_FROM_BACKEND)) {
          const backendAppId = await getAppId(cycleProps,
isDefaultTheme);
          if (backendAppId !== StatusResponse.ErrorConnection)
{
            config.appId = backendAppId;
          } else {
            // If fetching session from backend failed, we
must end processing here
            setError(true);
            return;
          }
        }
        const qlik = await getQlikDependencies(requireConfig,
setError);
        (qlik as any).theme.apply(
          isDefaultTheme
            ? "informaatiopalvelut-theme"
            : "informaatiopalvelut-turva-theme"
        );
        const appRes = await qlikAppLoader(config,
qlik).catch((err: Error) => {
```

```
          setError(true);
          setQlikAppError(err);
        });

        (appRes as any).on("error", (error) => {
          // Not sure if error is always an object:
          var outputError =
            typeof error == "object" ? JSON.stringify(error) :
error;
          //@ts-ignore
          window.appInsights?.trackException({
            exception: `qlik websocket error: ${outputError}`,
          });
        });
        setApp(appRes);
      })();
```

Code snippet 2. Loading of the Qlik sense app from server.

In the snippet provided, the code snippet commences by initially verifying whether the Qlik application has already been loaded. If not, it proceeds to retrieve the essential Qlik resources and ac- quire all necessary dependencies. In the event that an error is detected, the 'setError' variable is set to 'true,' triggering the initiation of an alternative process.

Next, establishing a connection to the Contentful API for retrieving texts in all supported languages (Finnish, English, and Swedish).

```
const useContenful = async (lang: Lang): Promise<any> => {
  const localEquivilent = {
    [Lang.en]: "en-US",
    [Lang.sv]: "sv-SE",
    [Lang.fi]: "fi-FI",
  };

  const client = createClient({
    space: "p8w932jpy7ij",
    //@ts-ignore
    accessToken: process.env.REACT_APP_CONTENFUL_ACCESS_TOKEN,
    host: "preview.contentful.com",
  });
  const returnObj = client
```

```
    .getEntries({
      limit: 1000,
      content_type: "resource",
      locale: localEquivilent[lang],
      select: "fields",
      "fields.application": "Yrityspulssi",
    })
    .then((response) => {
      let objectKeys: ContenfulKeyProp = {};
      Object.keys(ContenfulCodes).forEach((u) => {
        let so = [...response.items]
          .map((y) => y.fields)
          .find((v) => (v as any).key === ContenfulCodes[u]);
        if (so) {
          const { value } = so as any;
          objectKeys[u] = value;
        }
      });
      return objectKeys as ContenfulKeyProp;
    })
    .catch((error) => {
      return { error, errorFound: true };
    });

  return returnObj;
};
```

Code snippet 3. Fetching Contenful texts.

The above code snippet contains method that retrieves the codes that encompass all textual content in three languages: Finnish, Swedish, and English. In the event of a connection failure for any reason, it will generate an error, preventing the continuation of the Qlik app loading process.

Eventually, and important step was, creating the primary components of the mashup that required Qlik Sense connections, including object embedding, KPI data retrieval, filters, visualizations, and, most notably, Excel reports. This development work was undertaken in cooperation with the Qlik Sense developers on our team.

It has been also important to implement a mechanism designed to address sensitive case scenarios is paramount to safeguarding data from unauthorized portal users. The definition of these scenarios primarily relies on two Qlik variables that are transmitted to the frontend during the application loading process. Subsequently, these variables are processed within an object governed by a class known as 'PageAccessRights.' Data restrictions within this context can assume a variety of forms, specifically targeting sensitive features. For instance, certain users may be restricted from accessing specific graphs, or their usage of particular filters may be constrained.

Lastly, and as a main requirement of the project, there has been a need to create a specialized YRD reporting service tailored for Turva was undertaken at a later stage in the project. Turva, an insurance company, collaborates closely with Local-Tapiola, with the latter being responsible for managing Turva's IT services. The YRD reporting service designed for Turva mirrors the main version in terms of functionalities, mechanisms, and overall structure. The primary distinction lies in the customization of themes and colours to align with Turva's brand identity.

5.3 Deployment

The deployment process primarily encompassed two different environments within the parent application YRD Portal, namely testing and production. Both of these deployments were supervised by the team's dedicated DevOps engineer, utilizing the Azure Pipelines infrastructure for efficient management.

Deployment to testing was accomplished during the mid-stages of development and before high-level testing was launched. Both developers and testers from Fellowmind’s side and the customer side had access to the testing environment.

Then deployment to production was performed once all test cases have undergone thorough analysis and successfully passed through multiple stages of scrutiny, and was managed at the final stages of the implementation in September

2023. Subsequently, following the initial production deployment, minor adjustments were made to the application in response to some feedback received from customers. These adjustments aimed to enhance the user experience and ensure a more satisfying interaction with the application.

6 TESTING

6.1 Test Cases

Testing is an inherent component of our development process, typically conducted following each phase of development. Furthermore, comprehensive testing is undertaken from the customer's perspective. Effective and immediate coordination is consistently maintained among all stakeholders. Generally, the testing has been divided into three phases.

The first phase of testing was conducted from January 22, 2023, to March 1, 2023.

Table 2. First phase of test cases, their descriptions, and the results

Description of the test case	**Results**
The LocalTapiola's Qlik app loading in the frontend.	Pass. The Qlik app always connects successfully with the server and loads the Qlik app into the frontend.
Duet design part successful integration with the frontend. Meaning all components must behave as expected while browsing in terms of colours and themes of interactions with the component.	Pass. This task presented significant challenges, as initial tests failed due to the compatibility issues between Duet components and the version of the React application, leading to unexpected behaviors in certain components. However, the tests ultimately succeeded following a debugging process and the updating of several project libraries.
The filtering mechanism within the frontend should replicate the behaviour observed in the Qlik app, ensuring that alterations in filters accurately influence the corresponding Qlik app data. For instance, when filtering by year, the data displayed in the graphs within the Qlik objects should adapt in accordance with the selected filter criteria. Furthermore, certain filter options need	Pass after retest. The initial testing attempt encountered failure attributed to the mishandling of Qlik app page states. However, success was achieved subsequent to the proper management of Qlik app states.

to be dynamically disabled based on other selections. For instance, if the user selects the year 2017 and there is no corresponding car accident data for that specific year, the "Car accident" option within the "Type of accident" filter should be rendered inactive, preventing user selection due to its exclusion.	

Following the completion of the first phase, a second phase of testing was initiated, concentrating on more detailed functionalities. This phase was conducted from April 1, 2023, to June 1, 2023.

Table 3. Second phase of test cases, their descriptions, and the results

Description of the test case	**Results**
Download Qlik app graphs as an image from the frontend test.	Pass.
Download Qlik app graphs data in an exported Excel file.	Pass.
Transforming the graphical representations into tabular format by extracting data from the Qlik object hypercube and constructing a table utilizing a Duet table component.	Pass.
Maintaining consistent selections when switching between languages.	Pass after retest. The initial tests were unsuccessful, requiring adjustments to be made from the mashup perspective rather than the Qlik side to achieve functionality.
Retrieving the Qlik app ID from the backend, as opposed to relying on a hard-coded ID.	Pass.
Switching to Turva version of the YRD reporting service application.	Pass after retest. Transitioning to the Turva version necessitates adherence to Turva's established guidelines regarding themes, colours, and Duet components.
Accessibility according to WCAG.2.1 guideline. Retrieving the Qlik app ID from the backend, as opposed to relying on a	Pass. Accessibility held a para-mount significance within the specification. Compliance with

hard-coded ID.	accessibility guidelines encompassed colour schemes, visualizations, and ensuring full keyboard navigation capabilities for all features. Rigorous testing, including assessments with screen readers, was per-formed consistently.
Fetching all Contentful texts in the three available languages (Swedish, English, Finnish), ensuring that each Contentful entry contains the same texts in different languages. This process should occur prior to the loading of the Qlik application within the mashup. In the event of a connection error to Qlik, an error page should be displayed.	Pass. This has been tested many times and all tests went successful. Access key to make an API calls to the Contenful is set as an environmental variable.

The final phase of testing, conducted from September 1, 2023, to October 11, 2023, concentrated on sensitive case scenarios. This phase specifically addressed situations where certain users are restricted from accessing specific functionalities.

Table 4. Final phase of test cases, their descriptions, and the results

Description of the test case	**Results**
Restricted user from level A1 should not be able to access data of the accident page.	Pass.
Restricted user from level H1 should not be able to access data of the health page.	Pass.
Restricted user from level H1 should not be able to access data of the health page.	Pass.
Restricted user from level H2 should not be able to view all the Qlik graphs in the health page (still able to view the key performance indicators).	Pass.
Restricted user from level A3 should not be able to view some graphs in accident page.	Pass.
Restricted user from level H3 should not be able to view some graphs in health page.	Pass.
Language settings within the frontend	Pass.

should synchronize with changes made by the user in the YRD portal header. A component parameter is transmitted from the header to the reporting page, which can take one of three values (en, fi, sv) to define the language settings for the mashup and the entire YRD. The frontend should proactively detect any changes to the parameter value, thereby facilitating a change in the language.	The configuration has undergone multiple rounds of testing, all of which yielded successful results. The access key for making API calls to Contentful has been securely set as an environmental variable.
The system should retain selections made by the user even when the language is changed.	Pass.
The customer's name should be transmitted as a parameter from the YRD portal to the reporting service within the frontend.	Pass. The customer's name is employed in configuring the titles, ensuring that each downloaded Excel file or graph bears the customer's name in the title.

6.2 Improvements after Testing

Throughout the three phases of testing, enhancements were implemented to improve the efficacy of the applications and reduce the occurrence of errors perceivable by users. These improvements include:

- Enhancing the mechanism by which the mashup interacts with the Qlik server.
- In order to facilitate the downloading of Excel files, the xlsx library was replaced with exceljs, as testing had revealed that xlsx did not meet all specified requirements.
- The decision to employ the Duet table component, as opposed to a custom table, was made in response to feedback from the testing team, who expressed dissatisfaction with the custom table due to its failure to meet certain accessibility requirements.
- Generating internal hard-coded text entries within a JSON file to serve as a fallback in the event of a connection failure with the Contentful API.

- Employing the Contentful npm package library in lieu of Axios to facilitate API calls to the Contentful API prior to the mashup's loading.
- Enhancing error handling for all functions that return Promises.

7 SUMMARY

The primary objective of this project was to conduct an experimental exploration of delivering an analytical solution from an embedded Qlik application to a web-based application. The utilization of the QlikJS library was pervasive within the AngularJS framework. Initially, during the project's planning phase, the intention was to adhere to the framework supported by QlikJS, excluding React as a candidate. However, due to the expedient implementation of several high-priority specifications within the solution using React, a strategic shift occurred.

In response to this dynamic, the development of a custom QlikJS library has proposed, drawing inspiration from the existing one in AngularJS, given the shared foundation of JavaScript. To maintain adherence to the predetermined timeline, a series of coordination efforts and collaborations with the customer's team were essential. Nevertheless, numerous significant challenges and constraints were encountered throughout all phases of the project as described next.

Firstly, the changing requirements and specifications, as throughout the development process, the customer team continuously requested new alterations and enhancements. The confirmation of the final prototype design was not achieved until the fourth month, leading to numerous delays in the testing and delivery phases.

Secondly, the development process was marked by a high degree of interdependence among all team members. The progression of tasks for the Qlik mashup developer was contingent upon the guidance and tools provided by the Qlik Sense developer. The occurrence of overlapping holiday periods for team members had a discernible impact on the overall pace of progress.

Thirdly, the readiness of the development environment was a notable issue, as there was a long delay for some LT development tools to be ready for instance, the LT Azure machine and its setups for the development to proceed. Access to LT servers and services were not available during the initial stages.

Despite these constraints, we were able to adapt with them and find solutions or alternatives to any of the issues highlighted during the process. Objectives were set the beginning of planning and at the end they were all fulfilled.

The main achieved goals are presented next.

First and foremost, the success of the delivery was the most significant objective, as the embedded application has been formally transitioned into production, having met all the specifications outlined at the project's inception.

Also, compliance with WCAG was adhered, as the application has successfully undergone testing and demonstrated full compliance with accessibility regulations, marking a significant achievement as it represents the initial success in ensuring conformity with the essential and pertinent WCAG guidelines.

Embedding with YRD portal was also done as the integration of the YRD reporting service application with the YRD portal has been effectively achieved within a self-contained environment, permitting us to maintain the reporting application independently using the module federations method.

Additionally**,** there has been a usage of Duet library, as the integration of duet components has been seamlessly executed, aligning with the LocalTapiola's design prerequisites, and facilitating real-time design modifications from the design team within the reporting application.

Lastly, an important achievement is that, the Contentful API was effectively employed to retrieve and display all requisite textual content within the application, while affording the customer the autonomy to modify text elements in the future, across all three available languages: Finnish, Swedish, and English, without necessitating intervention from our team.

8 CONCLUSION

In conclusion, it is imperative to reflect on the overall execution and outcomes of the project. The project was well-organized and smoothly executed, following the expected steps with effective mentoring and periodic evaluations. Despite this, one must acknowledge a deviation from the initial plan, particularly concerning the project timeline. While the main goals of the project were indeed achieved, fulfilling the general purpose of the project's overview, the final delivery was not completed within the originally agreed-upon timeframe. Nevertheless, it is noteworthy that the application is now successfully in production.

Throughout the project, several challenges were encountered, each requiring innovative solutions. A significant issue was the evolving nature of requirements and specifications. To mitigate this, the team minimized resource allocation for integrating unconfirmed features and engaged in frequent stand-up meetings with the customer team to stay abreast of the latest information. Another obstacle was the delay in the readiness of the development environment. In response, the initial development phases were carried out in local computing environments, incorporating alternative resources and robust safeguards for sensitive LT data. Additionally, the challenge of team members working in different intervals was adeptly managed by recording any instances of a leave or an absence within a shared system, accessible to both Fellowmind and LocalTapiola. These solutions not only addressed the immediate challenges but also contributed to the overall success of the project and learning experience.

8.1 Future works and improvements

The YRD Reporting Service represents a significant milestone for Fellowmind, marking its involvement in the third major customer project. This initiative introduced the inaugural version of the YRD Reporting Service. Looking ahead to 2024, an ambitious plan for enhancements is in place, which includes expanding the ser-

vice to manage an increased volume of data from Qlik Sense, LocalTapiola's application. This expansion will not only encompass the addition of more pages but also focus on enhancing the performance and data processing speed of Qlik Sense.

Furthermore, the integration of new features and services is being explored, with a particular interest in leveraging AI technologies. The commencement of these improvements is projected to begin by May 2024, indicating a proactive approach towards continuous development and innovation in the service.

REFERENCES

[1] Microsoft website. Azure DevOps section. Retrieved 10/10/2023. https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure- devops?view=azure-devops

[2] React webpage. Main page. Retrieved 10/10/2023. https://legacy.reactjs.org/

[3] Webpack webpage. Main page. Retrieved 10/10/2023. https://webpack.js.org/concepts/

[4] Contenful website. FAQ section. Retrieved 10/10/2023. https://www.contentful.com/faq/about-contentful/#what-is-contentful

[5] Qlik sense website. Main page. Retrieved 10/10/2023. https://www.qlik.com/us/products/qlik-sense

[6] Module federation io webpage. Main page. Retrieved 1/11/2023. https://module-federation.io/docs/en/mf-docs/0.2/getting-started/

APPENDICES

APPENDIX 1

Information of secondary libraries used in the project.

About Duet design system.

Duet is a collection of reusable components and tools, guided by clear standards, that can be assembled together to build digital products for LocalTapiola and Turva.

The goal of Duet is to improve UI consistency and quality, while making our software design and development processes more efficient. Duet also helps to establish a common vocabulary between everyone in our organization and ease collaboration between different teams and disciplines.

The name “Duet” resembles the fact that there are two separate themes. One for LocalTapiola and one for Turva. “A duet” is a musical composition for two performers in which the performers have equal importance to the piece, often a composition involving two singers or two pianists.

Duet has a core team of designers and developers inside LocalTapiola who are dedicated to building and supporting the system.

In this project Duet has been utilized extensively in frontend development.

APPENDIX 2

Long code snippets and their explanations.

Class that is assigned to determine the sensitive case scenarios.

```
class PageAccessRights implements PageAccessPorps {
  public dataActionEnable: boolean = false;
  public sDataEnable: boolean = false;
  public showDimension: boolean = false;
  public showCoverGraph: boolean = false;
  public noCoverGraphText?: string | undefined;
  public yearQuarterSelection: boolean = false;
  public noDAEText?: string | undefined;
  public noSDEText?: string | undefined;
  public varReceived: string = "None";
  public keysTexts: ContenfulKeyProp = {};
```

```
public showHiddenYearBanner: boolean = false;
public showIka: boolean = false;
public excelButtonShow: boolean = false;
public hiddenYearText: string | undefined;
public timePeriod: string = "";
public insuredPersonBelow: string = "";
// Will be removed

constructor(
  keysTexts: ContenfulKeyProp,
  pageType: PageType,
  varReceived: string,
  timePeriod: string,
  inusredPersons: string | undefined
) {
  this.varReceived = varReceived;
  this.keysTexts = keysTexts;
  this.timePeriod = timePeriod;
  this.insuredPersonBelow = inusredPersons ?? "";
  if (varReceived === "None") {
    this.dataActionEnable = true;
    this.sDataEnable = true;
    this.showDimension = true;
    this.yearQuarterSelection = true;
    this.showCoverGraph = true;
    this.excelButtonShow = true;
    this.showIka = true;
    // TMP ACTION:
    // this.noSDEText = keysTexts.ypulssi_V03;
  } else {
    if (pageType === PageType.AccidentPage) {
      switch (varReceived) {
        case "A2":
          this.noDAEText = "ypulssi_V02";
          break;
        case "A3":
          this.dataActionEnable = true;
          this.sDataEnable = false;
          this.showIka = false;
          this.excelButtonShow = false;
          this.noSDEText = `ypulssi_V03||${v10Code}`
          break;
        default:
          break;
      }
    } else if (pageType === PageType.Health) {
```

```
        this.hiddenYearText = "ypulssi_V04" + "||" + this.in-
suredPersonBelow;
        switch (varReceived) {
          case "H2":
            this.noDAEText = "ypulssi_V02";
            this.dataActionEnable = false;
            break;
          case "H3":
            this.noDAEText = "ypulssi_V08";
            this.dataActionEnable = false;
            this.timePeriod = "";
            break;
          case "H4":
          case "H5":
            this.noSDEText =
              this.varReceived === "H4" ? `ypulssi_V03||${v10Code}`
: "ypulssi_V05";
            this.dataActionEnable = true;
            this.sDataEnable = false;
            this.showCoverGraph = true;
            this.showHiddenYearBanner = false;
            break;
          case "H6":
            this.dataActionEnable = true;
            this.sDataEnable = true;
            this.showHiddenYearBanner = true;
            this.yearQuarterSelection = false;
            break;
          case "H7":
            this.dataActionEnable = true;
            this.sDataEnable = true;
            this.yearQuarterSelection = false;
            break;
          default:
            break;
        }
      }
    }
  }

  public updateCondition = (qMatrix: NxCellRows) => {
    const selectionNum = Number(qMatrix[0].qText);
    const sensitiveNum = Number(qMatrix[1].qText);
    const coverNum = Number(qMatrix[2].qText);
    switch (this.varReceived) {
      case "H2":
```

```
  this.dataActionEnable = false;
  this.noDAEText = "ypulssi_V02";
  break;
case "H3":
  this.dataActionEnable = false;
  this.noDAEText = "ypulssi_V08";
  this.timePeriod = "";
  break;
case "H6":
case "H7":
case "None":
  // Case 1
  this.dataActionEnable = true;
  if (coverNum === 1) {
    this.noSDEText = "ypulssi_V06";
    this.sDataEnable = false;
    this.showCoverGraph = false;
    this.noCoverGraphText = "ypulssi_V06";
    this.yearQuarterSelection = true;
  } else {
    this.showCoverGraph = true;
    if (selectionNum >= 0 && sensitiveNum === 1) {
      this.sDataEnable = false;

      // Case H8
      if (selectionNum === 0) {
        this.noSDEText = "ypulssi_V05";
        this.sDataEnable = false;
      }
      // Case H9
      else if (selectionNum > 0) {
        this.noSDEText = "ypulssi_V06";
        this.sDataEnable = false;
      }
    }
  }
  if (this.varReceived === "None") {
    this.showHiddenYearBanner = false;
    this.yearQuarterSelection = true;
  }
  break;
case "H4":
case "H5":
case "None":
  // case 10
  if (coverNum === 1) {
```

```
          this.noSDEText = "ypulssi_V06";
          this.sDataEnable = false;
          this.showCoverGraph = false;
          this.noCoverGraphText = "ypulssi_V06";
          if (this.varReceived === "None") {
            this.yearQuarterSelection = true;
          }
        }

        break;
      default:
        break;
    }
  };
}
```

The provided code snippet illustrates a class that returns properties defining sensitive case scenarios for individual users. The constructor accepts various arguments, one of which is named 'varReceived.' This variable is received from the Qlik server during the loading process and is referenced in accordance with specific documentation available in the Confluence.