The implementation of Bifrostconnect Front-end scope, re-design and development with the relevant back-end support develop.

Fei Gu

January 3, 2024

Abstract

Contents

1 Introduction

1.1 The project background

This project marks the culmination of the EASV Computer Science program. Its primary objective is to showcase the breadth and depth of knowledge and skills that the student has amassed over two and a half years of rigorous study, and to apply these competencies in a practical setting.

In the scope of this project, the student will collaborate with a company to tackle a specific problem they are encountering. The student will leverage the knowledge and skills gleaned from their education to devise a solution.

This implies that the project will be directly relevant to the company's business domain, and will employ both software engineering and software development methodologies to identify, analyze, and ultimately resolve the problem.

1.2 About the company

in providing solutions for remote system access under stringent security conditions. The company's primary focus is on scenarios where the client's system lacks or is not permitted to have internet connectivity. Instead of relying on third-party services, BifrostConnect uses a relay device to facilitate remote access.

BifrostConnect, a startup founded in 2018, is a dedicated team of 10 to 12 professionals based in Copenhagen, Denmark.

Over the past five years, the company has developed a unique solution for remote system access. This solution is specifically designed for systems that operate under strict security conditions and are not permitted to have internet connectivity.

BifrostConnect's main offering is a hardware device coupled with a network application. This combination enables secure, remote access to client systems. In addition to this, the company provides related services and comprehensive solutions to cater to a wide range of client needs.

1.3 About the project

This project entails a thorough redesign and redevelopment of the existing front-end software, leveraging the foundation provided by the company's original software. Additionally, it involves enhancing the back-end to provide the necessary support for the new front-end.

2 Problem Statement

2.1 Statement

In the scope of this project, the integration of the Remote Access Interface and Tunnel Interface into the Device Manager Application is a primary objective. The goal is to create a single-page application that fully encapsulates the functionality of both the Remote Access and Tunnel Connection capabilities.

2.2 Situation

The existing BifrostConnect front-end application is divided into two distinct user interfaces: the Device Manager (DM) and the Remote Access Interface (RAI). This division requires users to manage their devices and users in one application, while operating their devices to use the Bifrost product in another web application. Moreover, the RAI is further split into two different user interfaces, known as the Classic RAI and Tunnel RAI. These interfaces direct users to different web applications depending on the type of access solution they wish to create.

This situation arose because the development of the Bifrost product was primarily focused on the device, with the application part not receiving as much attention. As a result, the application part was developed by different developers, each with their own design logic. This led to a lack of thorough planning for the application functionality during the development process. Consequently, the design logic of the product is not scalable, which has resulted in a diminished user experience and an exponential increase in code complexity, making it increasingly difficult to maintain and improve. Furthermore, the product design style lacks scalability due to a lack of modularity and code consistency, and the operation logic is unclear.

2.3 Potential Solution

To address those issue, there is a need to redesign and remake the front-end with a focus on implementing a modular unified front-end management system that encompasses all the necessary functionality.

The goal is to provide users with seamless control, monitoring, and access to their devices from anywhere, enhancing their overall user experience and productivity while keeping code complexity as low as possible.

For the implementation to be successful, we also need to have a back-end service, including an API and Database. The back-end server will contain the necessary service for the front-end to reach the purpose. Such as the RESTful API service, authentication service, MQTT service, Tunnel service and Database.

This project will follow the software design process. Therefore, it will also include the Software Development method and DevOps on all process as well.

This project is covered by NDA due to security and confidentiality reasons. Thus, the project will initially not be publicly deployed, however, I will described

the deployment in my report as well.

3 Requirement Analysis

3.1 Stakeholders

During this project. There are three stakeholders who are involved in this project. The user, the developer and the project owner.

3.1.1 User

The user is the person who will use the system. In this project, we can easily identify the user is the customer who will use the web application to manage their device and users, and create the RAI and Tunnel.

The user in this project are require the web application should be oprate easily, the features in this web application can be recongnized easily and the usage should be obivous.

3.1.2 Developer

The developer is the person who will develop, modify, scale and fix the system in any situation.

Therefore, the entire system should be easy to understand with the comments, easy to modify with the clear structure, easy to scale with reusable compoents and easy to identify the issue to fix any bugs.

3.1.3 Project Owner

The project owner is the person who will manage the project, in this case, the project owner is the company who own the product.

The project owner is require the project should be finished in time, can be coporeate with the hardware which be developed by the company, and the project can be easily to maintain, update, magration, and handover to other developers.

3.2 Scope

The scope of this project is to develop a web application which can be used to manage the device and users, and create the RAI and Tunnel.

Because the device manager already exist, the mean focus will be on the implementation of the RAI and Tunnel into the device manager. Which means we don't need to develop the device manager, also there are no necceary to develop the new features for the RAI and Tunnel interface.

Because the situation of the entire system was redudent and coupling, we have to re-design the system components and identify the relationship between them. The re-design of the system will be the main focus of this project.

Since the Device Manager is written in React framework, we have to use the same framework to develop the RAI and Tunnel interface.

3.3 Goals

The goal of this project is to develop a RAI and Tunnel interface in the Device Manager. The RAI and Tunnel interface should be able to create, modify, delete and view the access connection. Also can remote control the device and target equipment.

3.4 Bussines Domain

As we already discrbed about the bussines target and requirement about this project. We can identify the bussines domain model as following entitis:

Organization The organization have all the users and devices, which can set up the users and devices as a group. Therefore the organization have the following attributes:

- name: The name of the organization.
- users: which will be a list of users.
- devices: which will be a list of devices.
- groups: which will be a list of groups.

User The user can be the technitian or the oprator who is using the device to connect to the remote equipment. The user will be the end point which will start all of the process and end of the process as well. The user have the following attributes:

- username: which will identify the user.

Device The device is the hardware which will be connected to the remote equipment to allowed the user to control the remote equipment. The device have the following attributes:

- name: which will identity the device.

Group The group is the collection of the users and devices, which will be set up by the organization. Only the user and the device belong to the same group, the user can oprate the device, and only both the devices in the same group. The user can create the tunnel between the devices in the same group. The group have the following attributes:

- name: which will identity the group.
- users: which will be a list of users.
- devices: which will be a list of devices.

4 Software Design

4.1 Software Development Methods

4.1.1 Agile Software Development

The Agile software developement is a set of software development methods to aim to provide a sustainable, iterative and incremantal software developement process.

The Agile promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.

4.1.2 Unified Process

The unified software development process or unified process is an iterative and incremental software development process framework. This framework is based on the Agile increatemental and iterative software development process.

In this project, the unified process is used as the definitin of the software development process. The UP method such as the use case driven development, the iterative and incremental development are used in each phase of this project.

4.1.3 Feature Driven Development

"Feature Driven Development (FDD) is an iterative and incremental software development process." [1]

The FDD is a model-driven short-iteration process that consists of five basic activities:

1. Develop an overall model
2. Build a feature list
3. Plan by feature
4. Design by feature
5. Build by feature

4.1.4 Develop an overall model

The first step of the FDD is to develop an overall model. The overall model is a domain model that preposed to let the application can have a basic structure and can be run for no feature. After the overall model is developed, the feature list can be created for adding the features to the application.

In this project, the overall model is more like a pototype or demo of the application.

Front-end overall model At the front-end side, the overall model is a simple web page that can be used to show the basic structure of the application. Following the core concept of the React, the front end overall model will be only repersents the view of the application.

Back-end overall model At the back-end side, the overall model is data server which have the basic structure and a simple data model. The prepose of the backend overall model is not to provide any of the service or interface to eather the front-end or the database. It only use for the architecrure design to make sure the server side scalable and modular.

Database overall model The database overall model will indentify the bussiness domain of the application which is relavent to the device, user, and the data of the application.

4.1.5 Feature list

The feature list is a list of the features that the application will have. It will following the use-case that be indentify in the user story. And connect the all front-end, back-end and database together. Therefore, the each feature will produce the sequence start from the front-end to the back-end and then to the database, and return the result to the front-end to display to the user.

4.2 Technology selection

According to the requirments of this project. We need to build a full-stack B/S system with the database and other service.

4.2.1 Front-end

JAVASCRIPTl is the most popular language for the front-end development.

React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality. React Router is an example of such a library.

React-Router is a routing library for React. It abstracts away the details of server and client and allows developers to focus on building apps. It offers a simple, declarative API to let you build URL (Uniform Resource Locator) routing with ease. It supports server-side rendering and comes with routers for browsers and Node.js.

Redux is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger.

RTK is a powerful, opinionated Redux toolset for writing better reducers, organizing and accessing state in components, and managing side effects. It was originally created to help address three common concerns about Redux:

RTKQ is a powerful data fetching and caching tool. It is designed to simplify common cases for loading data in a web application, eliminating the need to hand-write data fetching & caching logic yourself.

TailwindCSS is a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

Semantic UI is a development framework that helps create beautiful, responsive layouts using human-friendly HTML. Semantic UI treats words and classes as exchangeable concepts. Classes use syntax from natural languages like noun/modifier relationships, word order, and plurality to link concepts intuitively.

Jest is a JavaScript testing framework maintained by Facebook, Inc. designed and built by Christoph Nakazawa with a focus on simplicity and support for large web applications. It works with projects using Babel, TypeScript, Node.js, React, Angular, Vue.js and Svelte. Jest does not require a browser to run tests, and it runs tests in parallel - this makes Jest fast. Jest is well-documented, requires little configuration and can be extended to match your requirements.

4.2.2 Back-end

C# is a general-purpose, multi-paradigm programming language.

ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications.

Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

4.2.3 Database

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.

4.2.4 Data communication

HTTP is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests. Though often based on a TCP/IP layer, it can be used on any reliable transport layer; that is, a protocol that doesn't lose messages silently, such as UDP.

Flatbuffers is an efficient cross platform serialization library for C++, C#, C, Go, Java, JavaScript, Lobster, Lua, TypeScript, PHP, Python, and Rust. It was originally created at Google for game development and other performance-critical applications.

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

webrtc is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.

WebRTC Datachannel is a component of WebRTC that enables peer-to-peer exchange of arbitrary data between two devices. It was added to the WebRTC standard by the World Wide Web Consortium (W3C) in January 2015.

4.2.5 DevOps

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GitHub Actions is an API for cause and effect on GitHub: orchestrate any workflow, based on any event, while GitHub manages the execution, provides rich feedback, and secures every step along the way. With GitHub Actions, workflows and steps are just code in a repository, so you can create, share, reuse, and fork your software development practices.

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

4.3 Domain model

From the bussines domain model, the entities and relation can be abstracted to the object domain model.

4.3.1 Entities

Entities The entities for this project are quite simple. There are only four entities in this project, The Organization, The User, The Device and The Group. But not like the bussines domain model only focus on the real world object exist in the bussines area. The object domain modal will more focus on the object type which will decide the object's attributes. And the relation will be determined by the object's method as well.

The user have the following attributes to describe it:

- id
- username
- email
- password
- role

The device have the following attributes:

- id
- name
- type

The group have the following attributes:

- id
- name
- users
- devices

The organization have the following attributes:

- id
- name
- users
- devices
- groups

4.3.2 Relations

The relation between the entities will following the bussines domain model. Following the logic of the all bussines area of the project. The first pair of relation is the user and the device: The user can oprating mutiple devices, but the device will only be operated by one user at a time. The group have a many to many relation with the user. The user can be in mutiple groups, and the group can have mutiple users. The group have a many to many relation also with the device. The device can be in mutiple groups, and the group can have mutiple devices. The organization have one to many relation with the user and the device. Which means the user and the device can only be in one organization. The organization can not share the user and the device with other organization as well. Eventually, the organization have a one to many relation with the groups.

4.4 Architecture design

4.4.1 overall architecture

In this project, the overall architecture design is based on C/S architecture, The client is a web browser. All the user interaction is done through the web browser. The uses operation include require data, display infomation, modify data, configure the device, create and oprating a classic remote access and tunnel will be happen in the web browser which is the frontEnd of the system.

5 Software Development

5.1 Front-end

5.2 Back-end

5.3 Database

5.4 WebRTC

5.5 MQTT

5.6 RESTful API

5.7 Authentication

6 Conclusion

6.1 Result

6.2 Discussion

6.3 Future Work

References

[1] Feature-driven development. Page Version ID: 1069092868.