Based on the real-life surveyed situation and the detailed system analysis in the chapter **??**, this chapter will center on the technologies used to develop the EPD Management System. This system comprises three main components: the Management Front-end, Back-end, and EPD Devices as the separated components. The Management UI is built using NextJS and TailwindCSS to provide an easy-to-use dashboard for managing multiple devices and data efficiently. The UI also incorporates protocols to connect to HTTP, MQTT servers, and EPD devices via Serial Port. All data is processed and stored on the server using MongoDB and ExpressJS. The server communicates with EPD devices through the MQTT protocol, using RabbitMQ as a broker and SSL/TLS to ensure a secure connection. The EPD devices connect to the broker and subscribe to a specific topic to receive and handle any information requested by users from the front-end side and send the status back to the broker.

0.1 Management UI (Front-end side)

0.1.1 NextJS

Next.js is a leading open-source JavaScript framework built on top of React, designed to simplify the development of single-page applications (SPAs) and server-rendered applications. As a React framework, Next.js enables the development of user interfaces using React components while offering additional features and optimizations. It stands out for its ease of use, performance optimization, and hybrid static and server rendering capabilities, contributing to improved user experiences and faster page loads. Next.js also supports automatic code splitting, ensuring that each page only loads the necessary JavaScript.

With its renowned Server-side Rendering feature (SSR) and other optimizations, Next.js significantly enhances website performance and responsive time while keeping the development task manageable and scalable. The framework is also equipped with tools for automatically measuring, tracking, and visualizing page performance metrics, such as Next.js Speed Insights and the useReportWebVitals hook, enabling developers to continuously maintain and improve websites' performance.

These performance benefits, including enhanced application performance with SSR and effective code management, have made Next.js a suitable choice for the project.

0.1.2 TailwindCSS

Tailwind CSS, a utility-first CSS framework, offers significant benefits for speeding up rendering times in web development. One of its primary benefits is its approach to generating the smallest possible CSS file for a project. The size of

CSS files directly impacts website loading speed. Larger CSS files take longer to load, leading to increased page load times. Slow-loading websites can suffer from higher bounce rates, reduced user engagement, and negative impacts on search engine rankings. Tailwind fixes this by only creating CSS for the components actually used in the project. Combined with minification and network compression, this results in remarkably small CSS files, often less than 10kB, even for large projects, which is essential for high performance.

In addition to reducing CSS file size, Tailwind CSS also enhances website accessibility and performance by reducing the amount of HTML code. This is achieved through the use of more efficient selectors, a core aspect of Tailwind's design philosophy. By optimizing the HTML and CSS of a website, Tailwind CSS helps improve both the speed and quality of the user experience.

Tailwind also contributes significantly to the improvement of website vitals. For instance, the First Contentful Paint (FCP) metric, which is crucial for understanding perceived load speed, can see a reduction of up to 36% compared to other styling methods like styled components. Such improvements in Speed Index and Largest Contentful Paint metrics indicate that Tailwind CSS not only accelerates development time but also enhances the performance and responsiveness of the final product.

0.2 Server (Back-end side)

Because of the complexity of the system that needs both processing data from the client side and also communicating with EPD devices, this Server section is divided into two main parts: API server handling user's requests and storing data in MongoDB database, and RabbitMQ broker communicating with devices and sending data back to ExpressJS server to process.

The two servers used in the system are dedicated servers hosted in Hetzner, a prominent web hosting provider and data center operator in Europe. Hetzner is well-known for its robust data center infrastructure, a wide range of hosting services, and commitment to connectivity and performance. Hosted in Hetzner's dedicated servers, the system can benefit from the single allocation of full CPU, RAM, and storage resources that are not shared with other users. It also improves performance, security, and the scalability of the system significantly and enables developers to fully control and customize for specific needs.

0.2.1 API Server

This part of the back-end server is responsible for receiving user requests, relaying data to the MQTT Server, and storing data in the system. The server is written

in the MVCS design pattern for code transparency and management and leverages **ExpressJS**, a minimal and flexible Node.js web application framework, to help enhance the server's capabilities. It enables efficient API creation and middleware integration, streamlining development and improving maintainability.

To store and secure user data, the back-end system uses **MongoDB Community Edition for Linux** (hereafter referred to as '**MongoDB**'), which stands out from other database solutions due to its distinct advantages in storing and processing data. MongoDB is adept at handling vast amounts of user and device data, storing it securely across distributed systems, in this case, the *clusters*. This design enhances data redundancy and security and allows for horizontal scaling, accommodating large-scale data without compromising system performance. In tandem with **ExpressJS**, this also ensures efficient, high-performance query and transaction processing promptly while keeping the development and maintenance processes streamlined and minimal.

Utilized for API design and documentation in this project, Swagger from OpenAPI, known as OpenAPI Specification (OAS), offers a comprehensive, detailed description of the API structure, including endpoints, operations, and parameters. This documentation tool not only aids in API development and testing by allowing for interaction with API resources without implementing logic, but it also streamlines client SDK generation across various programming languages. Leveraging Swagger documentation in the project helps speed up the development cycle and enhance efficiency in identifying bugs and issues during testing, ultimately resulting in a more robust and user-friendly application.

0.2.2 MQTT Broker

To enable real-time communication and data transfer between users and devices, the system also operates as an MQTT broker, facilitating the exchange of information between parties. This is acquired by using RabbitMQ, an open-source message broker software that enables applications to communicate with each other and exchange information efficiently. It acts as an intermediary for messaging by accepting and forwarding messages, making it a critical tool for handling asynchronous communication between EPD devices and the server. Its reliability and scalability make it a preferred choice for enterprises needing to ensure message delivery without loss, even in high-throughput scenarios. RabbitMQ's ability to decouple processes also leads to more resilient and manageable application architectures.

0.3 EPD device

0.3.1 ESP32-C3 Supermini

The EPD devices in the project run on ESP32-C3 Supermini, a compact, general-purpose Wi-Fi and Bluetooth LE module inheriting the features from ESP32 microcontrollers. The ESP32-C3 Supermini uses ESP32-C3, a 32-bit, RISC-V-based MCU with 400KB of SRAM, capable of running at up to 160 MHz, as its core. It also integrates 2.4 GHz Wi-Fi and Bluetooth 5 (LE) with long-range support. It also features 22 programmable GPIOs (General Purpose Input/Output), supporting a variety of interfaces, including ADC (Analog to Digital Converter), SPI (Serial Peripheral Interface), UART (Universal Asynchronous Receiver/Transmitter), I2C (Inter-Integrated Circuit), RMT (Remote Control), etc. The ESP32-C3 Supermini includes a Type-C USB, 4MB of Flash, and supports 12x IO, which is sufficient for most of the development tasks in the project.

0.3.2 E-paper display

The display panel uses the Waveshare 2.9-inch e-Ink Display Module, which is a versatile and efficient display solution for various applications. It features a 2.9-inch screen with a resolution of 296x128 pixels and supports partial refresh, allowing for low power consumption and a wide viewing angle. The display presents content in black and white with a 4-level grey scale, providing a paper-like effect that is ideal for applications such as price tags, shelf labels, and industrial instruments. It is connected and communicated with the ESP32-C3 Supermini module via the SPI interface, which is particularly notable for its ultra-low power consumption, making it suitable for this system where power efficiency is crucial.

0.4 PlatformIO

PlatformIO is a powerful and versatile open-source ecosystem widely used for IoT and embedded systems development. Its cross-platform compatibility allows integration with Visual Studio Code, offering flexibility for developers. PlatformIO stands out for its extensive support for numerous microcontrollers and development boards, including Arduino, ESP8266, ESP32, and STM32, catering to a diverse range of projects. One of the most renowned features of PlatformIO is its comprehensive suite of development tools, including a robust library manager that simplifies the incorporation and management of external libraries and a unified debugger that supports a wide range of debugging tools. It also supports multiple development environments, which is particularly valuable in complex development scenarios, enabling developers to create versatile applications that can be easily adapted and deployed across different hardware platforms. Last but not least, PlatformIO

has also raised a vibrant community around it, enriching the platform and providing extensive documentation, active user forums, and shared knowledge, making it an invaluable tool for both hobbyists and professionals in embedded systems development.

With the aid of PlatformIO, the project has significantly advanced in both development efficiency and technical robustness. It provides a comprehensive toolset for managing, developing, and testing the development boards, which is also integrated with various development environments, making the development and debugging process easier.

0.5 Cloudflare

Cloudflare is a widely recognized internet services company that specializes in providing a range of solutions aimed at enhancing online security and performance. Renowned for its expansive infrastructure designed to handle significant web traffic volumes, Cloudflare serves a diverse clientele, from individual website owners to large enterprises, ensuring their online presence is secure, fast, and reliable. It also provides free SSL/TLS as part of its range of security and performance services for websites, designed to be easily implemented and managed, providing robust encryption for data in transit between the user's browser and the web server.