



# **BME CAPSTONE DESIGN COURSE**

## **PROJECT REPORT**

### **TeleUroflow**

A web-based app for visualization and  
control of Uroflowmeter

#### **Instructor(s)**

Prof. Vo Van Toi

#### **TA(s)**

Nguyen Le Y  
Vo Minh Thien  
Le Thi Thuy Tien

#### **Group 1**

<b>Name</b>	<b>Student's ID</b>
Huynh Nguyen Minh Tri	BEBEIU20257
Chau Thanh Huy	BEBEIU20018
Pham Hoai Bao	BEBEIU20175

Ho Chi Minh City  
**February 2024**

## Table of Contents

<b>List of Figures .....</b>	<b>3</b>
<b>List of Tables .....</b>	<b>4</b>
<b><i>Abstract</i>.....</b>	<b>5</b>
<b>I. Introduction.....</b>	<b>6</b>
<b>A. Background.....</b>	<b>6</b>
<b>B. Project objectives and motivation.....</b>	<b>7</b>
<b>C. Literature Review.....</b>	<b>8</b>
<b>D. Design requirements.....</b>	<b>14</b>
<b>II. Methodology .....</b>	<b>15</b>
<b>A. Design Overview .....</b>	<b>15</b>
<b>B. Hardware Description.....</b>	<b>18</b>
<b>C. Software Description .....</b>	<b>21</b>
<b>D. Final Prototype .....</b>	<b>33</b>
<b>III. Testing and Performance Evaluation .....</b>	<b>35</b>
<b>A. Testing plans .....</b>	<b>35</b>
<b>B. Performance Evaluation .....</b>	<b>35</b>
<b>C. Product Economic Evaluation.....</b>	<b>41</b>
<b>IV. Project Management .....</b>	<b>43</b>
<b>V. Conclusion and Discussion.....</b>	<b>45</b>
<b>VI. Acknowledgment.....</b>	<b>46</b>
<b>VII. References.....</b>	<b>47</b>
<b>VIII. Appendices.....</b>	<b>49</b>

## List of Figures

<b>Figure 1.</b> The prevalence of voiding dysfunction in women [2].....	6
<b>Figure 2.</b> The ICS recommended nomenclature for Uroflowmetry data [8].....	7
<b>Figure 3.</b> Cloud hosting services for managing and storing data.....	9
<b>Figure 4.</b> Google BigTable field concept [27]. ....	10
<b>Figure 5.</b> Wireless Uroflowmeter – model DanFlow 3000 – developed by Innologic.....	10
<b>Figure 6.</b> Advin Uroflowmeter System can be configured to the PC via USB connection. ....	11
<b>Figure 7.</b> Laborie wireless Uroflowmeter accuracy and utilizes Bluetooth to connect to PC. ....	13
<b>Figure 8.</b> Finished hardware device. ....	15
<b>Figure 9.</b> TeleUroflow Login Interface.....	15
<b>Figure 10.</b> TeleUroflow Patient Monitoring Dashboard.....	16
<b>Figure 11.</b> Block diagram of the full system.....	17
<b>Figure 12.</b> Load cell circuit. ....	18
<b>Figure 13.</b> Real time circuit and power circuit.....	18
<b>Figure 14.</b> Hardware circuit schematic. ....	18
<b>Figure 15.</b> Complete circuit. ....	19
<b>Figure 16.</b> Complete device. ....	19
<b>Figure 17.</b> Flow chart of machine. ....	22
<b>Figure 18.</b> Flow chart of data pipeline. ....	27
<b>Figure 19.</b> Patients table.....	27
<b>Figure 20.</b> Signals table.....	28
<b>Figure 21.</b> Time series signal table. ....	28
<b>Figure 22.</b> FAST API for backend. ....	31
<b>Figure 23.</b> Flow chart of web application. ....	32
<b>Figure 24.</b> TeleUroflow Login Interface.....	33
<b>Figure 25.</b> TeleUroflow patient information dashboard .....	33
<b>Figure 26.</b> TeleUroflow measuring dashboard .....	34
<b>Figure 27.</b> Fetch patient information. ....	38
<b>Figure 28.</b> Fetch signal metadata. ....	38
<b>Figure 29.</b> Fetch time series signal. ....	39
<b>Figure 30.</b> a) Add patient b) Modify patient. ....	39
<b>Figure 31.</b> a) Delete drop down box b) Delete pop up window.....	39
<b>Figure 32.</b> Activate signal. ....	40
<b>Figure 33:</b> Delete signal.....	40
<b>Figure 34.</b> Hosting pricing calculation performed by AWS cost analysis software.....	41
<b>Figure 35.</b> ETL pricing calculation performed by AWS cost analysis software.....	42

## List of Tables

<b>Table 1.</b> TeleUroflow specifications. ....	14
<b>Table 2.</b> Detailed function of the system's components .....	17
<b>Table 3.</b> List of components and materials for hardware device.....	19
<b>Table 4.</b> Decision matrix comparing different database engine options. ....	22
<b>Table 5.</b> Confusion matrix comparing different APIs used to integrate the DBMS with the web software. .	23
<b>Table 6.</b> Confusion matrix for ETL services.....	25
<b>Table 7.</b> Decision matrix for backend programming languages. ....	29
<b>Table 8.</b> Decision matrix for frontend programming languages. ....	29
<b>Table 9.</b> Database update latency trial. ....	36
<b>Table 10.</b> Comparison of signal before and after processing.....	36
<b>Table 12.</b> Price list of main components on the circuit. ....	41
<b>Table 13.</b> Project timeline and tasks. ....	43
<b>Table 14.</b> Member and task divisions. ....	44

## ***Abstract***

**Problem** – Uroflowmetry (UF) is guideline-recommended tool for treating individuals with lower urinary tract symptoms and benign prostatic hyperplasia (BPH/LUTS). Traditionally, this test is performed in a clinical setting on an outpatient basis, where patients are required to urinate into a uroflowmeter, while maintaining privacy. This process is considered unnatural, demanding patients to urinate "on command," often after excessively filling their bladder. This can lead to variable results due to the stress experienced by patients under these conditions, known as paruresis or shy bladder syndrome, resulting in data that inaccurately represent normal urinary function. As a result, multiple tests are advised, necessitating several visits to a clinic, which can be time-consuming and costly. Consequently, there's a push for the creation of novel uroflowmetry methods that are user-friendly, convenient, and portable for home use.

**Methods** – In recent years, telemedicine and telehealth emerged as crucial strategies to alleviate the burden on healthcare systems and to provide continuous patient care through remote screening, diagnosis, and follow-ups. In this project, we introduced an IoT integrated solution that combines a previously developed Uroflowmeter with a database system to tackle these issues effectively. Our goal was twofold: firstly, to explain the benefits of using an affordable Uroflowmeter design that captures tele-uroflowmetry data accurately and reliably; and secondly, to develop a web-based platform that not only collects and securely stores this data in a scalable database but also enables remote viewing and analysis by healthcare professionals at central hospitals.

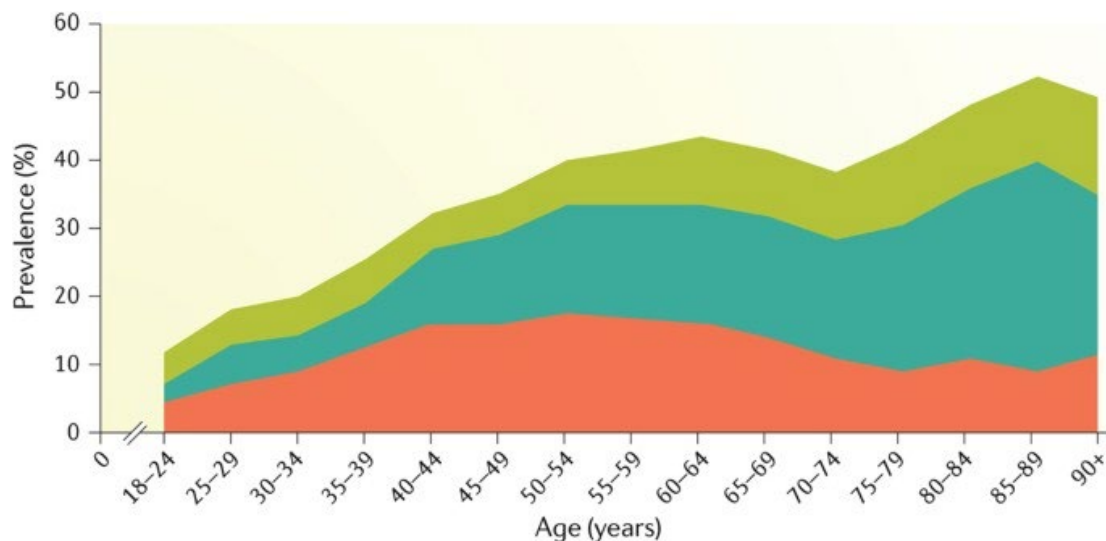
**Results** – We developed a web-based platform to control and visualize UF signal, namely TeleUroflow, and successfully connected the Uroflowmeter to our platform. Preliminary testing showed rapid and accurate data transmission onto the software with average transmission time from 0.2-0.3 seconds. Signal was uploaded onto the platform using REST API and PostgreSQL engine and smoothed using weighted moving average method. Finally, other UF parameters were derived from raw data on backend and saved with signal-patient ID. Nevertheless, the project has reached the crucial part of prototype for interfacing the Uroflowmeter with the web-based software, while future research is required to enhance the system security before being used for clinical practice.

**Keywords** — **TeleUroflow, Uroflowmeter, Telemedicine, IoT, Web-based software.**

# I. Introduction

## A. Background

Voiding dysfunction, a prevalent issue associated with aging, significantly affects the quality of life, particularly in males over 60, impacting more than 60% of this demographic [1]. It is approximated that 4 billion people worldwide currently suffer from urinary tract symptoms that negatively affect quality of life and incur significant cost [2], [3]. Traditional clinical evaluation involves invasive urodynamic testing, which requires catheterization in clinic or hospital settings, limiting assessments to periodic intervals [4]. Addressing this issue, Uroflowmetry has emerged as a common and non-invasive physiological approach for evaluating lower urinary tract obstruction. This method has proven effective in offering objective evidence regarding the extent of prostate gland enlargement, overactive bladder, urinary incontinence, or neurogenic bladder [5]. Uroflowmetry evaluates urinary tract function by measuring the speed of urine flow, the volume expelled, and the duration of the process.

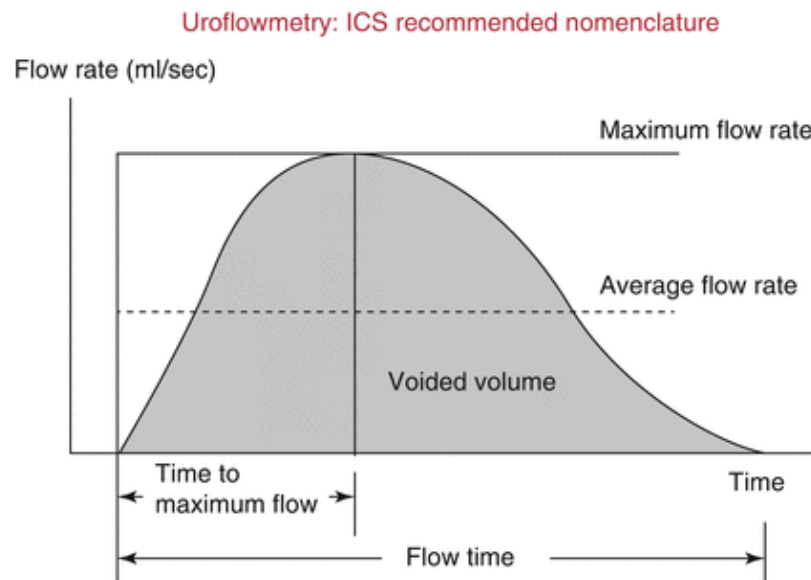


**Figure 1.** The prevalence of voiding dysfunction in women [2].

*Green: urgency incontinence, Blue: mixed incontinence, Red: Stress incontinence.*

Presently, Uroflowmetry (UF) is conducted in outpatient settings within designated facilities, requiring individuals to urinate into a uroflowmeter. The 2018 International Continence Society (ICS) Standardization Report outlines necessary conditions for UF, including sufficient privacy, a natural urge to urinate, and the need for multiple, accurate tests [6]. Measurements should capture a graphical representation of flow rate over time, peak urinary flow rate ( $Q_{max}$ ), total volume voided ( $V_{void}$ ), and the volume remaining after voiding (post-void residual volume, PVR). Yet, the conventional approach to free urodynamic testing (FUT), typically done once in a clinical setting, struggles to accurately reflect natural urination patterns and exhibits significant variability between patients. The requirement for immediate urination, often with an overly full bladder, introduces considerable inconsistency in results due to stress-related effects on urine flow, known as paruresis or shy bladder syndrome [7]. Despite the time and financial burden, multiple readings in a clinical environment have been the norm to circumvent these challenges and enhance the precision of

diagnoses. Consequently, it is advised to perform several tests to achieve reliable data, acknowledging the investment of time and resources required for both healthcare providers and patients making frequent visits to the clinic.



**Figure 2.** The ICS recommended nomenclature for Uroflowmetry data [8].

A UF device designed for home use, capable of conducting numerous measurements aligned with an individual's daily voiding patterns, has the potential to be employed in the evaluation of men experiencing Lower Urinary Tract Symptoms (LUTS) [9]. Such an approach not only circumvents the limitations of office-based assessments but also harnesses the statistical advantage of averaging multiple measurements and aligning with circadian rhythms. Yet, the practical deployment of such devices has been minimal, and the adoption of online telemonitoring systems is still emerging. On the other hand, the COVID-19 crisis has highlighted the critical need for telemedicine options, prompting the European Association of Urology to recommend its application for the management of male LUTS and benign prostatic hyperplasia (BPH) now and in the future [10]. Innovatively, the launch of smartphone apps for uroflowmetry, which employ unique algorithms to assess the sound of urination, presents an inventive alternative to traditional approaches. This progress allows for the seamless integration of uroflowmetry data into electronic health records, streamlining the process and reducing costs for both patients and healthcare providers. Conventionally, uroflowmetry in clinics can present numerous operational and clinical challenges for urologists, including patient discomfort and the logistical burden of conducting repeat tests, which increase costs and demand on staff time. Hence, a user-friendly, portable, and home-based UF system could address the shortcomings of clinic-based measurements.

## B. Project objectives and motivation

In this project, our primary objective was to develop a web-based platform for storing and visualizing UF data through IoT and connect the previously developed Uroflowmeter to the platform. As the Uroflowmeter was developed with an ESP32 Wifi module, the UF signal after measurement remotely

will be transmitted to the web-based system for storage, management, and interpretation by medical doctors at the central hospital on a daily basis. In brief, our project can be divided into four main objectives:

- Develop a fully functional web-based software and database for data storage, the software will include user-friendly interface and authorization to ensure the users' privacy, and documentation of the system architecture, design, and implementation
- Collect, transmit the UF signal from the Uroflowmeter to the database using the Wifi module of ESP32, ensure rapid and accurate data transmission
- Preprocess and process the load signal to obtain sufficient UF parameters according to the ICS requirements, and include normograms for interpretation and comparison with normal healthy UF graphs
- Test the signal processing workflow and the system performance

## **C. Literature Review**

### **a. Existing methods**

To allow tele-monitoring of UF signal at the patients houses on a daily basis, it is quint-essential that the platform can handle a large magnitude of data while ensuring the data privacy. Once the signal is acquired from the ESP32 device, it needs to be transmitted to a database engine for storage and retrieval. This process involves choosing the right database management system (DBMS), data visualization libraries, database integration and API tools, and deployment methods. Each component plays a pivotal role in ensuring the application's functionality, performance, and user experience.

#### **Database Management Systems (DBMS)**

When it comes to storing structured data, MySQL stands out as a highly scalable and reliable open-source relational database management system (RDBMS) [11]. Its widespread use and support for defined relationships between entities make it an excellent choice for applications requiring structured data storage. Alternatively, PostgreSQL offers advanced features such as data integrity constraints and user-defined data types, according to the PostgreSQL Global Development Group [12]. Its flexibility and extensibility make it well-suited for complex data modeling and analysis, providing a robust foundation for applications dealing with intricate data relationships. For applications requiring the storage of unstructured or semi-structured data, MongoDB, as detailed by MongoDB, Inc., presents a compelling option [13]. This NoSQL database stores data in flexible JSON-like documents, facilitating scalability and ease of data handling, especially for evolving data models [14].

#### **Data visualization libraries**

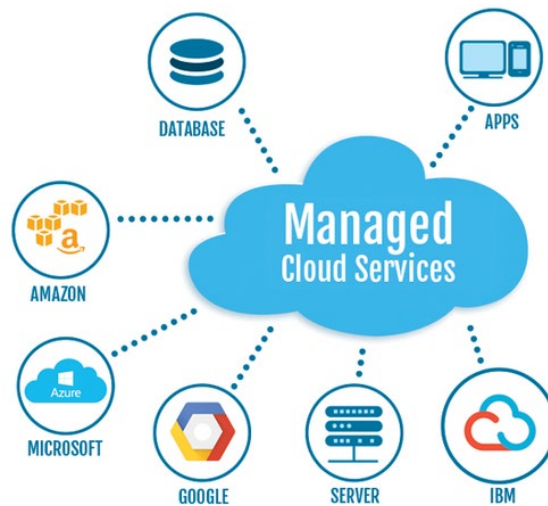
The visualization of uroflowmetry signals is integral to the application's effectiveness. Libraries such as D3.js offer powerful capabilities for creating custom data visualizations, allowing for highly customized visuals that can communicate complex data patterns and trends [15]. For developers



seeking a more straightforward approach, Chart.js provides an easy-to-use platform for creating common chart types like line charts, bar charts, and pie charts [16]. Its higher-level abstraction is ideal for quick data visualization needs. Highcharts, a commercial option, offers high-quality and interactive data visualizations with a comprehensive suite of chart types and advanced customization options, ensuring cross-browser compatibility and rich user experience [17], [18].

### Deployment & storage format options

Deployment is the final step in making the web application accessible to users. Cloud hosting services like Amazon Web Services (AWS) and Microsoft Azure provide scalable and cost-effective deployment solutions [19], [20]. Self-hosting offers greater control over the infrastructure but requires significant technical expertise [21]. For static web applications, platforms such as Netlify and Vercel offer ideal solutions [22], [23]. Integrating the ESP32 device with the web application and the chosen database engine requires careful planning around communication protocols, data formats, and security measures. Standardizing the data format (e.g., JSON, CSV, XML) exchanged between these components is essential for ensuring compatibility and seamless communication.



**Figure 3.** Cloud hosting services for managing and storing data.

Many contemporary data-processing platforms, such as Google's BigTable, Hadoop's HBase, and Hypertable, adopt a Key-Value approach to restructure their databases for efficient data storage [24], [25], [26]. This method deconstructs each data row, diverging from the conventional database architecture, and grants autonomy to each data field. This innovation not only eliminates the redundancy associated with null values but also promotes the characteristics of distributed data and scalability.

In the case of BigTable, data is organized into three primary fields: Row, Column, and Timestamp, with each data set referred to as a Cell, as illustrated in **Figure 3**. All data is consolidated into a single table, albeit with unique configurations for these three fields [27]. The management and storage of data are facilitated by three distinct keys: the Row key, which assesses data load; the Column key,

which oversees storage and retrieval operations; and the Timestamp key, which archives the versions of data at various points in time.

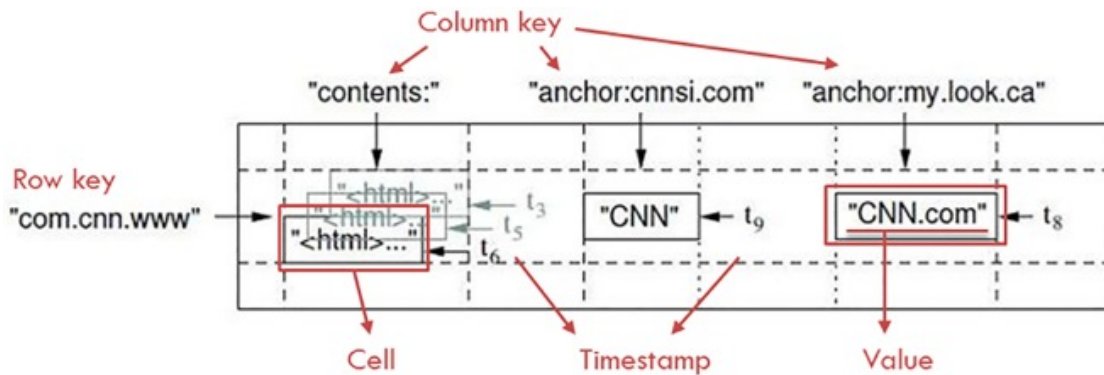


Figure 4. Google BigTable field concept [27].

## b. Existing devices/applications on markets

### Danflow 3000 – Innologic

The DanFlow 3000 is the flagship Uroflowmeter developed by Innologic, Australia's leading innovation in digital wireless uroflowmetry. The DanFlow series, including the 3000 model, revolutionizes uroflowmetry with its cutting-edge weight cell transducer technology. The device sets itself apart by offering a completely wireless experience, eliminating the need for cumbersome wires and constant power supply within the testing environment. Its base unit, powered by a durable lithium battery, liberates the uroflowmetry process from traditional constraints, allowing for greater flexibility in choosing the location for testing—be it various hospital areas or even between different sites, with an impressive working distance of up to 50 meters. The device allows truly wireless operation, markedly enhancing the distance between the flow meter and the printer using Bluetooth technology. Unlike conventional Bluetooth-connected devices, the DanFlow 3000 operates on the 866 MHz frequency, ensuring superior range, signal stability, and uninterrupted data transmission through walls and doors. Moreover, the DanFlow 3000 can be connected seamlessly to any computer or notebook via a wireless USB dongle. It accommodates both male and female patients with its two height-adjustable stands, tailored for men who prefer standing during the test and easily adjustable for women and pediatric patients for seated testing.



Figure 5. Wireless Uroflowmeter – model DanFlow 3000 – developed by Innologic.

## Advin PC Based Uroflowmetry System

The Advin PC Based Uroflow, an advanced iteration of Uroflowmetry Systems, is engineered to meticulously monitor urinary volume and flow rate. This system distinguishes itself by integrating directly with a computer or laptop via a USB connection, ensuring that all reports are stored on the PC for seamless access and management. One of the primary advantages of this PC-based Uroflowmetry System is its capacity to efficiently store and retrieve patient reports and information on a computer or laptop. This feature facilitates effortless recall and comparison of graphs, streamlining patient monitoring and analysis. The Advin PC Based Uroflow embodies the pinnacle of modern uroflowmetry technology as a fully automatic, microprocessor-controlled device. Its operation is governed by a digitally controlled weight-based flow transducer, ensuring precision and reliability in measurements. Employed to diagnose urinary tract deficiencies, the Uroflowmetry System offers a non-invasive method for daily flow studies, proving indispensable in urological diagnostics. Recognized under various names such as Urine Flow Meter, Computer Based Urinary Flow Meter, and Uroflowmetry Equipment, this system is renowned for its comprehensive functionality.



**Figure 6.** Advin Uroflowmeter System can be configured to the PC via USB connection.

### Key advanced features include:

1. **Data Management:** Direct storage of reports and patient information on a PC or laptop, with easy retrieval and report comparison capabilities.
2. **Digital Control:** Utilizes a digitally controlled weight-based mechanism for enhanced accuracy.
3. **Fully Automated:** Features automatic start and stop functions, alongside automatic analysis for user convenience.
4. **Accuracy and Safety:** Delivers accurate results through a non-invasive method, ensuring no risk to the patient and eliminating the need for hospitalization.
5. **Cost-Effectiveness and Efficiency:** Offers a cost-effective solution for uroflowmetry testing, characterized by its lightweight, compact design. It's easy to handle, requires no maintenance, and aligns with international quality standards.

Moreover, its design ensures ease of operation, cleaning, and reusability, marking the Advin PC Based Uroflow as a leading solution in the field of uroflowmetry, designed to meet the needs of modern healthcare practices.

### **Laborie Urocap™ IV Wireless Uroflowmeter**

The UROCAP™ IV offers a blend of portability and advanced wireless capabilities. This device, empowered with Bluetooth® technology, provides quick and accurate measurements of urinary flow and volume, making it an indispensable tool for both routine flow studies and a variety of clinical applications. Its versatility is further enhanced by the option to use it as a standalone device or integrate it with the Goby™ UDS system, catering to diverse clinical needs.

Key attributes of the UROCAP™ IV elevate its utility and user experience. The device's utilization of Class 1, long-range Bluetooth® wireless technology significantly enhances patient privacy during the voiding process. Its battery-powered design frees users from the constraints of power cords, facilitating effortless mobility and use in various clinical settings. Additionally, the UROCAP™ IV is designed with a custom-molded, water-protected frame, not only ensuring device durability but also simplifying hygiene practices with its IP54 rating, making it an ideal choice for busy clinic environments.

The compact and portable nature of the UROCAP™ IV does not compromise its functionality. Equipped with VBN software, it allows for playback and thorough analysis of test data. Its design prioritizes user-friendliness with features like procedure voice-guidance and a customizable interface, simplifying its operation. The device also supports patient diagnosis through its nomogram features and offers extensive reporting capabilities via integration with I-LIST®, enhancing clinical workflow and patient management. Furthermore, its compatibility with Hospital Information Systems (HIS) or Electronic Medical Records (EMR) aids in creating a paperless clinical environment, streamlining data management and accessibility.

Vietnam, with its large and aging population, faces a significant challenge with the prevalence of urological conditions among its elderly. Without timely and proper monitoring, these conditions can become life-threatening. Our team's market research revealed a lack of simple, user-friendly devices for patients to self-monitor at home. Despite the abundance of uroflowmeter options in the medical market, produced by various international manufacturers using different technologies, most are not designed with portability or ease of self-administration in mind, necessitating frequent doctor visits. Furthermore, the high cost of these foreign-made products underscores the need for a domestically produced alternative, offering similar capabilities at a more accessible price point, thereby improving access to healthcare services. Vietnam's ongoing enhancements to its telecommunications infrastructure place the country in an advantageous position to adopt digital health solutions. The telemedicine project, still in its nascent stages, promises to bridge the gap towards a comprehensive online hospital network.



**Figure 7.** Laborie wireless Uroflowmeter upholds portability, accuracy and utilizes Bluetooth to connect to PC.

Despite the abovementioned innovative approaches to facilitating remote healthcare, the existing products has yet to streamline communication between patients and doctors, often requiring the manual transfer of data via social media. These aspects present our project with a unique opportunity to surpass existing Uroflowmeter in the market by addressing these shortcomings.

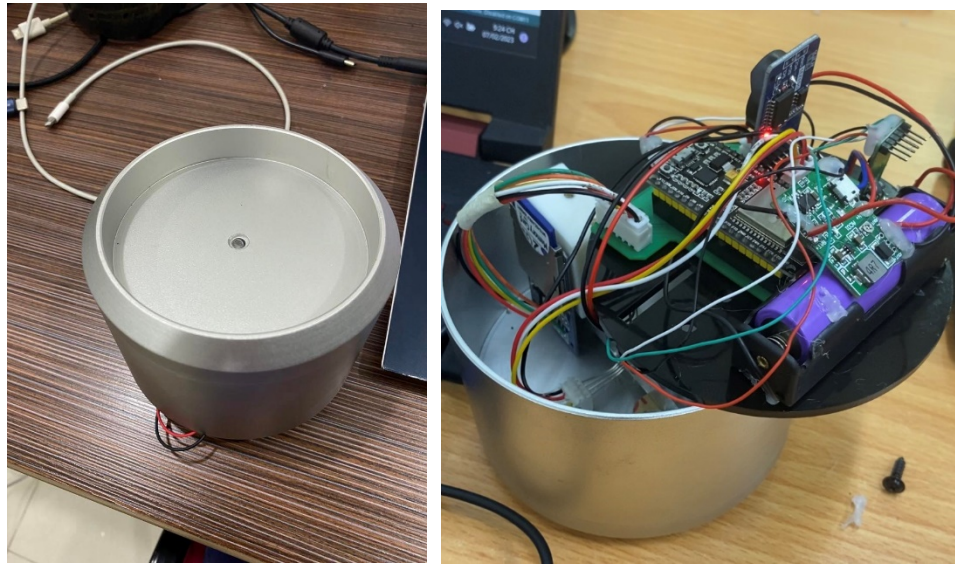
## D. Design requirements

**Table 1.** TeleUroflow specifications.

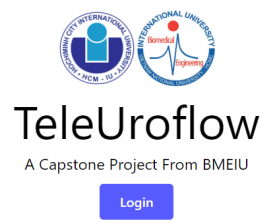
<b>Specifications</b>	<b>Requirements</b>
<b>Connection</b>	Through ESP32 Wifi module, ideally should allow measurement from distance ~20m Fast and easy connection of the Uroflowmeter and the platform via Wifi
<b>Software compatibility</b>	PC, Windows, Android, IOS
<b>Reliability</b>	Design the system to be robust and resilient to failures, minimizing downtime and data loss
<b>Transmission time</b>	Less than 5 seconds
<b>Signal processing and analysis</b>	Preprocess, smooth raw data for better visualization Ensure data accuracy and avoid time delay or shift during transmission or processing Provide tools for in-depth data analysis to obtain basic UF parameters
<b>Authorization</b>	Require log in before entering the software Unique account and password for different user Master account for developer High-level account for clinical practitioners
<b>Functionality</b>	Patient info (patient ID, name, birthdate, gender, signal ID) Graphs: volume-time, flow rate-time, nomogram UF parameters: median volume, voided mixed volume, max flow rate, average flow rate, time to max flow rate
<b>User interface</b>	User friendly, bright color tone (light blue or green) Ensure accessibility, especially for people with color blindness Language: Vietnamese, English (preferable)

## II. Methodology

### A. Design Overview



**Figure 8.** Finished hardware device.



**Figure 9.** TeleUroflow Login Interface

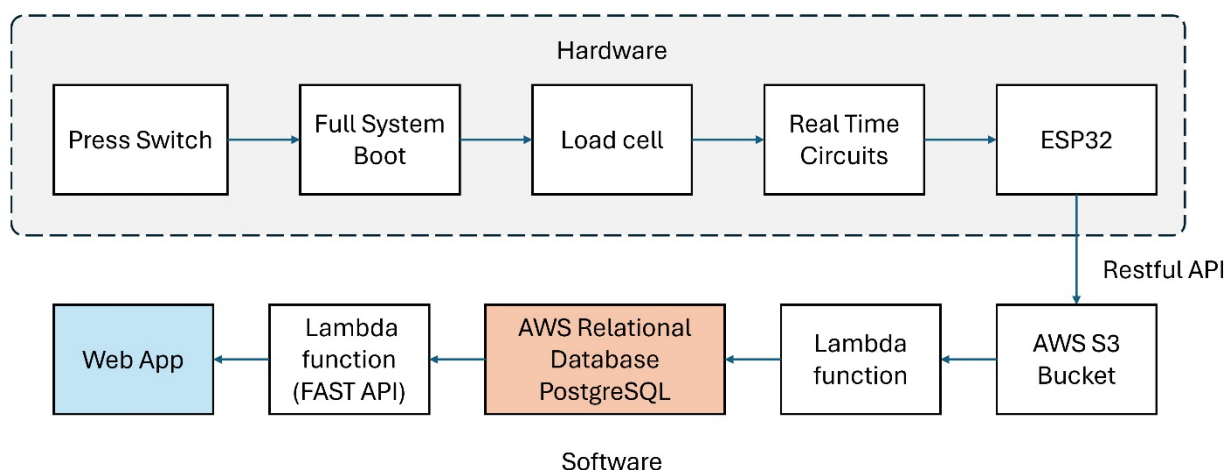




Figure 10. TeleUroflow Patient Monitoring Dashboard

In order to concentrate on the development of databases and websites (**Figure 9 – 10**) for urinary devices, we obtained permission from Mr. Y to inherit all hardware components from the previous Capstone group (**Figure 8**). Prior to its delivery, the device had been subjected to numerous hardware enhancements by the previous Capstone team. The aforementioned enhancements comprised the ability to retrieve readily available components and the simplicity of substituting components in the event that maintenance was necessary. Notably, in an effort to reduce connectivity between components, they deliberately relocated the connecting cables, resulting in a more streamlined design than the initial incarnation of the uroflow meter. To offer a full overview of the system, **Figure 11** shown below is the block diagram defining the overall design.





**Figure 11.** Block diagram of the full system.

Our tele-urinary measuring system begins with a user activating a press switch, triggering a full system boot. The load cell then measures the urinary output, with real-time circuits processing the data swiftly. This data is transmitted via an ESP32 microcontroller to the cloud using a Restful API.

In the cloud, the data is first stored in an AWS S3 Bucket. It is processed by a Lambda function and subsequently stored in a PostgreSQL database on AWS. Another Lambda function, utilizing FAST API, retrieves this data for the web application, which then presents it to the user. This sequence ensures that urinary measurements are accurately captured, processed in real-time, and made accessible for monitoring through a user-friendly interface.

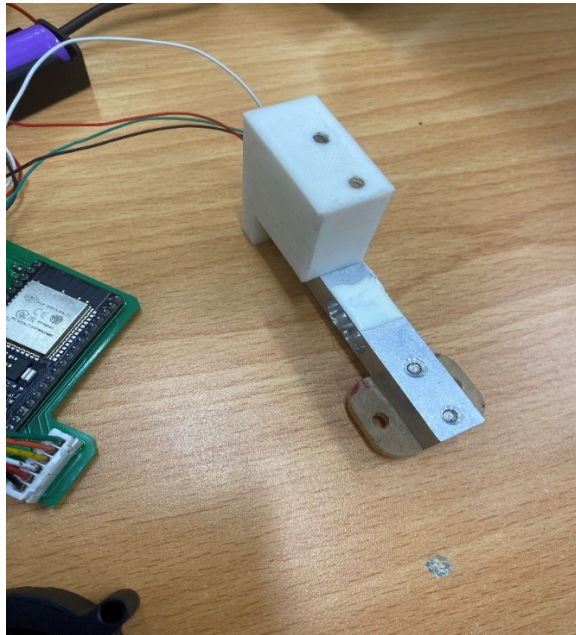
For more information, the features of each component of our system are shown in depth in the table below.

**Table 2.** Detailed function of the system's components

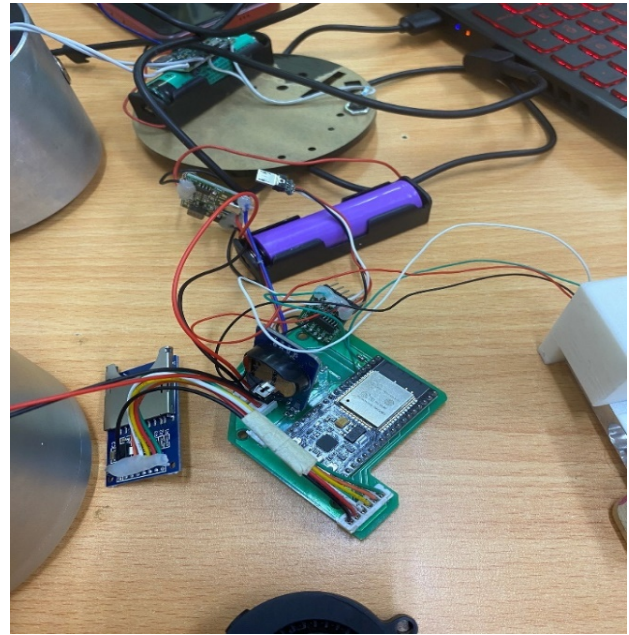
	Component	Function
<b>Hardware</b>	Press Switch	User activates the measurement process.
	Full System Boot	Initiates the startup sequence for all hardware components.
	Load Cell	Measures the volume of urinary output.
	Real-Time Circuits	Handles immediate processing of data from the load cell.
	ESP32	Microcontroller that sends data from real-time circuits to the software via a Restful API.
<b>Software &amp; Services</b>	AWS Restful API	Interface for transferring data from hardware to cloud services.
	AWS S3 Bucket	Initial data storage service that receives data from ESP32 through the Restful API.
	Lambda Function	Processes uroflowmetry signal stored in the AWS S3 Bucket.

	AWS Relational Database PostgreSQL	Durable storage service for structured data after processing by the Lambda function.
	Lambda Function (FAST API)	Handles requests from the Web App, retrieves data from the PostgreSQL database, and sends responses back.
	Web App	The user interface that displays processed data to end-users and may allow for further interaction.

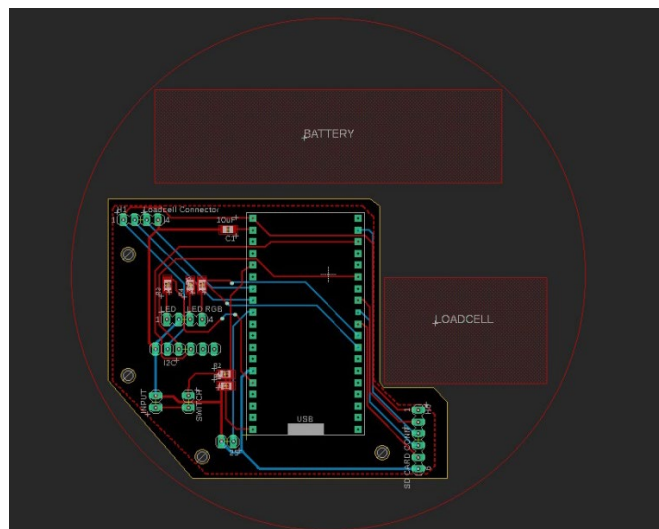
## B. Hardware Description



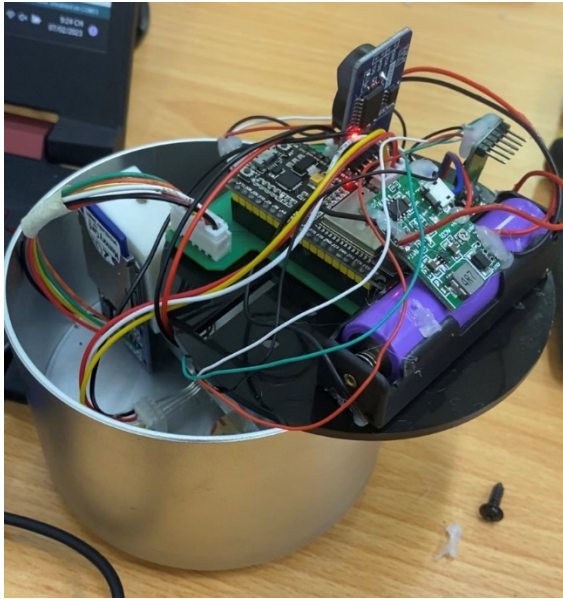
**Figure 12.** Load cell circuit.



**Figure 13.** Real time circuit and power circuit.



**Figure 14.** Hardware circuit schematic.



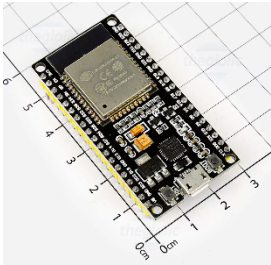
**Figure 15.** Complete circuit.



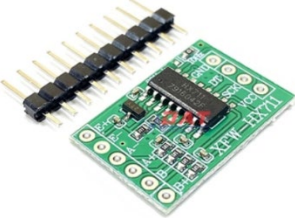
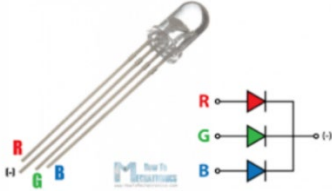

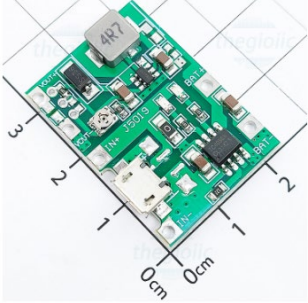
**Figure 16.** Complete device.

Upon a detailed examination, the device's hardware comprises a variety of components, as outlined in **Table 3**. The previous team's report was thoroughly reviewed, but it did not contain a specific decision matrix for comparison. Despite this, each type of component has been meticulously chosen, as confirmed by that team. They have ensured that the components listed in the table above were selected with careful consideration, meeting the criteria of both price and performance. These selections adhere to the pre-established standards of hardware, demonstrating a balance between cost-effectiveness and optimal performance.

**Table 3.** List of components and materials for hardware device.

Components	Specifications
 <p>ESP32 NodeMCU LuaNode32 38 Pin Wifi Transceiver Module</p>	<p>Type: Wifi + Bluetooth Module  Model: ESP32 38 pins  Power supply voltage (USB): 5V DC  Input/Output Voltage: 3.3V DC  Power consumption: 5<math>\mu</math>A in suspension mode  Performance: Up to 600 DMIPS  Frequency: up to 240MHz  Wifi: 802.11 B/g/n/E/I (802.11N @ 2.4 GHz up to 150 Mbit/S)  Bluetooth: 4.2 BR/EDR BLE 2 control modes  Memory: 448 Kbytes ROM, 520 Kbytes SRAM, 6 Kbytes SRAM on RTC and QSPI Flash/SRAM chip support  USB-Serial Chip: CP2102  Antenna: PCB  Digital GPIO: 24 pins (some pins only as inputs)  Digital Analog: 12bit SAR type ADC, supports</p>



	<p>measurements on up to 18 channels, some pins support an amplifier with gain programming.</p> <p>Security: IEEE 802.11, including WPA, WPA/WPA2 and WAPI</p> <p>Hardware Accelerated Cryptography: AES, SHA-2, RSA, Ellipse Cryptographic Curve (ECC), Random Number Generator (RNG)</p>
 <p>ADC Converter Circuit 24bit Loadcell HX711</p>	<p>Operating voltage: 2.7~5VDC</p> <p>Consumption current: &lt; 1.5 mA</p> <p>Sampling rate: 10 - 80 SPS (customizable)</p> <p>Resolution: 24-bit ADC</p> <p>Voltage resolution: 40mV</p> <p>Dimensions: 38 x 21 x 10 mm</p>
 <p>RGB Led Driver Circuit</p>	<p>Use voltage: 5VDC.</p> <p>Number of RGB LEDs: 1 RGB 5050 led</p> <p>Dimensions: 15 x 10.6mm</p>
 <p>Real Time Circuit RTC DS1302</p>	<p>Main IC: RTC DS1302</p> <p>Power supply: 3.3/5VDC (depending on the communication voltage level of the device to be connected).</p> <p>Interface: 3-Wires Interface (Reset, Data, Clock)</p> <p>Store and provide real-time information: day, month, year, hour, minute, second, ...</p> <p>There is a backup battery to maintain time in case of power failure.</p> <p>Dimensions: 44 x 23mm</p>
 <p>18650 Battery Charger Circuit 1 Battery 1S With Protection 1A</p>	<p>Input voltage: 4.5-8 VDC</p> <p>Output voltage: 4.3-27 VDC</p> <p>Charging voltage: 4.2 VDC</p> <p>Charging current: Up to 1A</p> <p>Discharge Current: Max. 2A</p> <p>Dimensions: 3.3 x 2.3cm</p>

## C. Software Description

### a. Coding framework for hardware

In terms of the embedded code, we have inherited the Uroflowmetry machine from Mr. Y, along with the accompanying Arduino code. This code is responsible for the machine's functionality, which includes recording real-time signals and saving them to an SPIFFS, the inner storage of ESP32. However, this functionality does not fully meet the requirements of our project, as our objective is to transition towards telemedicine, where data is not stored locally on the machine but rather sent to the cloud. Consequently, the addition of internet connectivity and the capability to transmit data to the cloud are necessary.

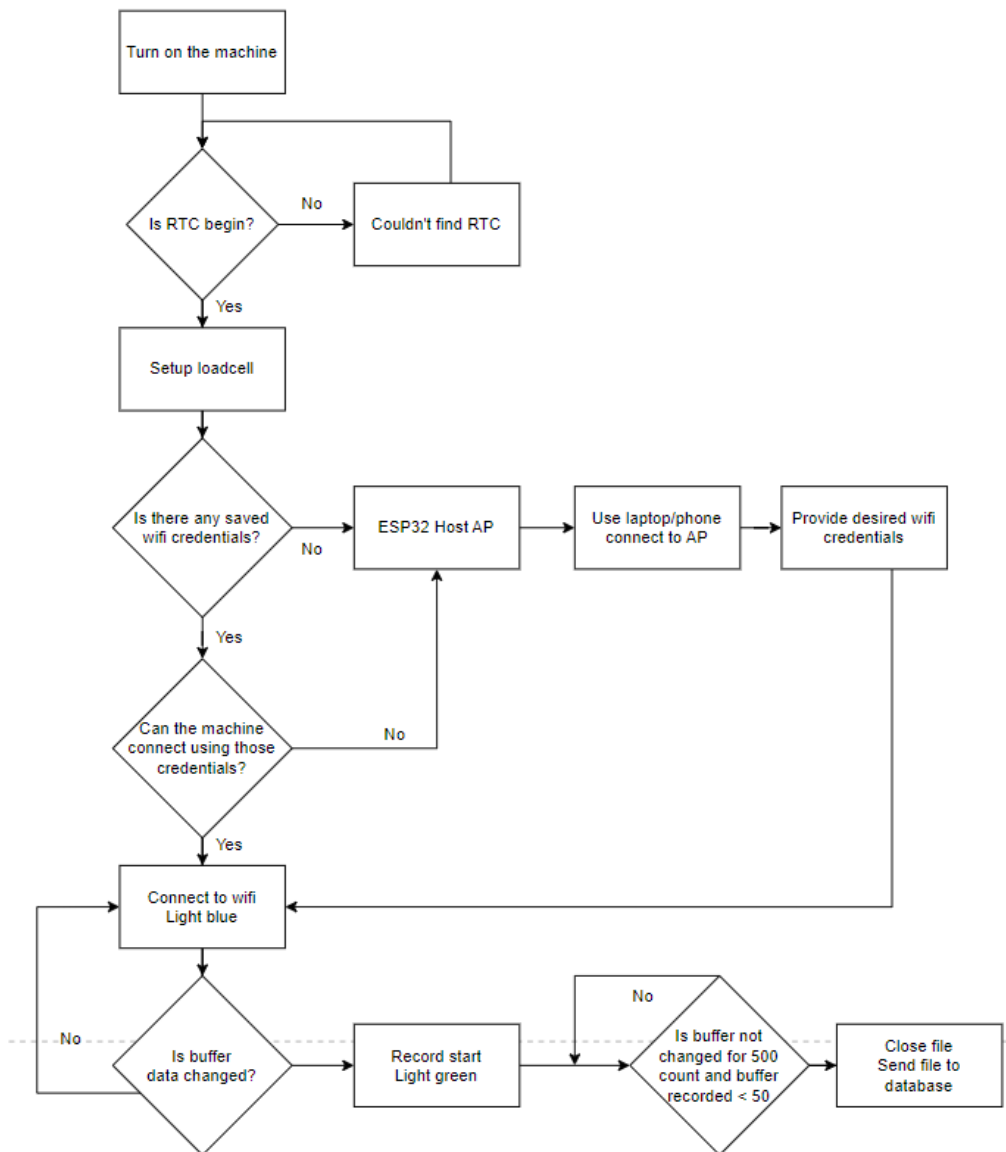
Regarding internet connectivity, hard coding the machine is not a feasible solution, necessitating alternative methods for data transmission. An initial idea was to utilize Bluetooth, connecting the machine to a laptop to facilitate data transmission via the ESP32. However, this approach was deemed unsuitable due to our objective of transmitting data to the cloud. Moreover, given the inherent characteristics of Uroflowmetry, where measurements are performed in a restroom, connecting the machine to any type of device is both redundant and inconvenient. Therefore, a Wi-Fi connection would be more ideal and appropriate for our case.

In terms of Arduino libraries, the Wi-Fi Manager is most commonly used for Wi-Fi connections, particularly for initial setup. The principles of the Wi-Fi Manager as well as the logic of recording urine signal will be elucidated in the flow chart (**Figure 17**).

The flow chart commences with the activation of the machine. At this stage, the system verifies whether the Real-Time Clock (RTC) has started. If not, the system will continue to search for the RTC until it is located. Once the RTC is found, the load cell is set up, and a Wi-Fi connection check is performed.

The system then checks for saved credentials. If such credentials exist, the system will attempt to use them for the ESP32 to connect to Wi-Fi. If the connection cannot be established, or if there are no saved credentials, the ESP32 will host its own Access Point (AP). Users can then connect another device to this AP and provide the desired Wi-Fi credentials to assist the ESP32 in establishing a Wi-Fi connection.

Upon successful connection to Wi-Fi, the indicator light turns blue, and the buffer begins to detect weight. If a significant change in weight is detected, the recording process commences, and the indicator light turns green. The recording continues until there is no change for 500 counts, and the recorded buffer at that moment is less than 50 (the weight of the container). At this point, the file is closed and sent to the database. This process ensures efficient and accurate data collection and transmission in our telemedicine application.



**Figure 17.** Flow chart of machine.

## b. Database management system engine and data pipeline

**Table 4.** Decision matrix comparing different database engine options.

Feature	MongoDB	MySQL	PostgreSQL	Firebase
Data model	NoSQL (document-oriented)	Relational	Relational (document-oriented)	NoSQL (document-oriented)
Data scalability	Highly scalable			
Performance	Fast for read and write operations			
Data consistency	Eventual consistency	Strong consistency	Strong consistency	Eventual consistency

<b>Data security</b>	Supports encryption and access control			
<b>Ease of use</b>	Relatively easy to learn and use	More complex to learn and use	Easy to learn and use	Relatively easy to learn and use
<b>Cost</b>	Open source	Open source	Open source	Free for small projects, paid plans for larger projects
<b>Use cases</b>	Web applications, mobile applications, IoT applications	Web applications, e-commerce applications, enterprise applications	Data warehousing, analytical applications, complex data models	Web applications, mobile applications, gaming applications

The primary objective of our project is to integrate a Uroflowmetry machine with a database system and a web application for data visualization. The selection of an appropriate Database Management System (DBMS) is crucial for this purpose. Among the various DBMS engines available, PostgreSQL has emerged as the most suitable candidate, as demonstrated in our decision matrix. This is due to its hybrid data model, which supports both relational and document-oriented structures, allowing us to design our own database with our own uses, which is described later in this section. In addition, PostgreSQL matches or surpasses other DBMSs in terms of security, consistency, cost, and ease of use, where it is the easiest and most consistency when compared to other candidates.

The integration of ESP32 with PostgreSQL presents a complex problem that necessitates the implementation of several intermediary steps to establish a comprehensive data pipeline. This pipeline is underpinned by the principles of Extract, Transform, Load (ETL) processes. However, to achieve the ETL the data needs to be sent into a raw zone, since currently there is no direct ETL services that integrate directly with ESP32. Therefore, we have to make connection with some storage, which helps to connect ESP32 with the storage for storing the raw signal. One of the ways to achieve this is using REST API. REST API an acronym for Representational State Transfer API, defines a set of guidelines for building web services that ensure they are interoperable, scalable, and reliable. This architectural style leverages HTTP, the foundational protocol facilitating server-client communication on the web. Employing REST APIs offers several advantages, including interoperability, scalability, and reliability. Interoperability is achieved through adherence to a universally recognized set of principles, enabling seamless interaction among diverse applications across various programming environments and platforms. The stateless nature and layered system of REST APIs contribute to their scalability, allowing them to efficiently manage increased traffic volumes. Furthermore, the reliability of REST APIs is anchored in their use of HTTP, a protocol that has been extensively tested and validated over time.

**Table 5.** Confusion matrix comparing different APIs used to integrate the DBMS with the web software.

<b>Feature</b>	<b>REST API</b>	<b>GraphQL</b>	<b>SOAP</b>	<b>gRPC</b>
<b>Protocol</b>	HTTP/HTTPS (Widely supported)	HTTP/HTTPS	HTTP, HTTPS, SMTP, TCP, and more (More	HTTP/2 (Limited browser support)

			complex setup)	
<b>Data Format</b>	JSON, XML (Versatile and widely used)	JSON (Efficient but less versatile)	XML (Verbose and heavier)	Protocol Buffers (Efficient but requires learning curve)
<b>Design Philosophy</b>	Resource-centric, stateless (Intuitive and flexible)	Query language for API, precise data fetching (Requires more initial setup)	Action-oriented, heavily standardized (More rigid)	Contract-first (High efficiency but less flexible)
<b>Performance</b>	Good, adaptable to various needs	Optimized data fetching (May require more complex queries)	Slower due to XML verbosity	High (May be overkill for simpler applications)
<b>Complexity</b>	Lower, easy to start with and scale	Higher, due to query complexity	Higher, due to strict specifications	Moderate, with additional overhead for defining data structures
<b>Flexibility</b>	High, with support for multiple data formats and methods	High, but demands more from the client to shape responses	Lower, due to strict standards	High, but closely tied to service definition
<b>Use Cases</b>	Broad range, especially CRUD operations and general web services	Complex queries and mobile apps (where precise data is crucial)	Enterprise and legacy systems (where standards are critical)	High-performance and microservices (specific scenarios)
<b>Security</b>	Standard web security methods, easy to implement	Requires careful consideration for query complexity	Built-in standards like WS-Security	Strong authentication and encryption (Require setup)

**Table 1** compares some APIs used to integrate the DBMS with our developed web software. By comparison, REST API is the preferred choice to the other options. Its protocol and data format are widely supported and versatile, offering a balance between ease of use, performance, and flexibility. REST API's intuitive design, coupled with its scalability and the straightforward implementation of security measures, makes it particularly suitable for a wide range of use cases, from simple CRUD operations to more complex web services. It strikes an optimal balance, providing sufficient flexibility for most applications without the complexity and overhead associated with other API types. With this selection, the next step is choosing storage to store the raw signal recorded from ESP32. Among many storages, AWS S3 Bucket is chosen, due to its ability to integrate directly with Restful API, as well as its connection with many ETL services. Some of the most popular ETL



services is described in decision matrix below.

**Table 6.** Confusion matrix for ETL services.

Criteria	AWS Lambda	AWS Glue	AWS Data Pipeline	Amazon Redshift	Amazon EMR
<b>Data volume per uses</b>	Medium	High	High	High	Very high
<b>Cost</b>	Based on number of execution time and trigger	Based on data processed	Based on data processed and compute resources used	Based on storage and compute resources used	Based on number of instances and instance types
<b>Flexibility</b>	High (custom code tailored to specific needs)	Low (fully managed service)	Medium (some setup required)	Medium (SQL-based transformations)	High (have ability of distributed systems)
<b>Real-time processing</b>	Excellent (Immediate processing upon data upload)	Good (Can be scheduled or event-driven)	Good (Can be scheduled or event-driven)	Poor (Batch-oriented)	Poor (Batch-oriented)
<b>Integration with AWS S3</b>	Excellent (Direct integration with S3 events)	Good (Can read/write from S3)			
<b>Custom Metric Computation</b>	Excellent (Full control over code)	Good (Uses PySpark, Scala, or Python)	Medium (Limited by pre-defined activities)	Medium (SQL-based transformations)	Good (Uses Hadoop, Spark)
<b>Integration with PostgreSQL</b>	Excellent (Can use PostgreSQL RDS directly)	Medium (Requires additional ETL to move data)	Medium (Requires additional ETL to move data)	Poor (Not designed for direct database integration)	Poor (Not designed for direct database integration)
<b>Serverless Architecture</b>	Excellent (No server management needed)	Excellent (No server management needed)	Fair (Some resources may need management)	Poor (Requires cluster management)	Poor (Requires cluster management)
<b>Time limit</b>	15 minutes	None	None	None	None

As delineated in **Table 4**, a multitude of services exist that facilitate the Extract, Transform, Load (ETL) process. However, for the scope and requirements of this project, the AWS Lambda function emerges as the most suitable choice, and this preference is substantiated by several reasons.

Firstly, the AWS Lambda function's capability to integrate directly with S3 and PostgreSQL, without necessitating additional ETL to transfer data, is invaluable. This feature simplifies the setup of the data pipeline and keeps the process as streamlined as possible.

Secondly, the high flexibility of the AWS Lambda function is a significant advantage. It provides the ability to handle time series signals as CSV-type data through custom metric computation. This is facilitated by Python, one of the most user-friendly programming languages, thereby making it an essential factor in this decision.

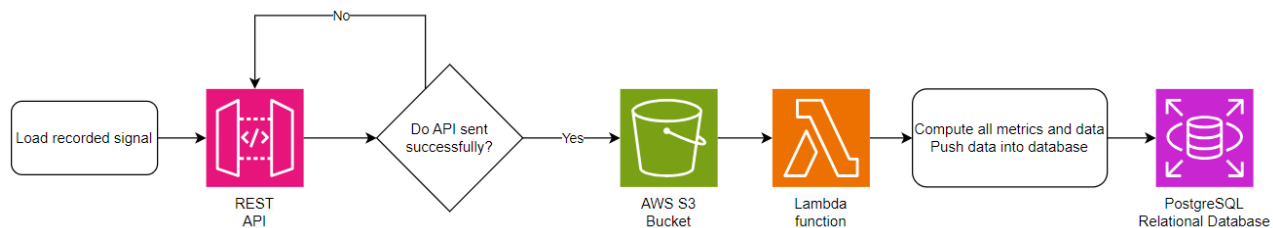
Thirdly, the serverless architecture of the AWS Lambda function offers a substantial benefit. The absence of server management requirements saves considerable time, allowing the project team to focus on other critical aspects of the project. Furthermore, the serverless architecture eliminates the need for round-the-clock server hosting, which would otherwise incur continuous cloud costs. Instead, payment is made only upon triggering, thereby reducing unnecessary expenditure.

Finally, despite its limitation concerning data volume per use, the AWS Lambda function is well-suited for this project. The natural characteristics of the time series signal for uroflow occupy no more than 1GB. Therefore, the AWS Lambda function meets this requirement and helps avoid redundant costs for a data volume limit that will never be reached. Also, about the actual cost, the economic evaluation section will be analyzed more, where Lambda function cost nothing when assume that we have 1000 patient per day. Moreover, the time limit is not a concern, as this trigger will last under one second, a detail that will be carefully highlighted in the testing plan section of this report.

With the selected ETL and DBMS services, the data pipeline process can be delineated as follows:

- The initiation of the data pipeline is marked by the successful recording of a signal by the ESP32.
- Subsequently, the recorded signal is loaded, and a request is dispatched to the RESTful API.
- This sequence of actions is reiterated until the data is successfully transmitted to AWS S3.
- Upon the successful upload of new data to AWS S3, the Lambda function is triggered. This function is designed to instantaneously load the new data, compute, and process all requisite metrics.
- The processing step of lambda function employs the weighted moving average and monotonicity enforcement to rectify load cell measurement fluctuations. This ensures a consistent volume increase, eliminating the possibility of a negative flow rate.
- The metrics computed by the Lambda function encompass the flow rate per second, median volume, voided volume, maximum flow rate, average flow rate, time to maximum flow rate, and flow time.
- Upon completion of these computations, the Lambda function compiles a data frame that encapsulates all the computed information, and subsequently pushes this data frame into PostgreSQL.

This process ensures a seamless and efficient data pipeline from the ESP32 to PostgreSQL, leveraging the capabilities of AWS services to handle the extraction, transformation, and loading of data. This methodology provides a robust solution to the challenges inherent in the direct integration between ESP32 and PostgreSQL, thereby facilitating efficient data management and analysis.



**Figure 18.** Flow chart of data pipeline.

This project aims to utilize the relational characteristics of PostgreSQL for database design, with a specific focus on uniquely identifying signal data corresponding to individual patients. The database comprises three tables:

- **Patients:** This table contains basic information about each patient, including:
  - **Patient ID:** A unique identifier for each patient, defined by the Unix timestamp at the moment the patient is added to the database. This serves as the primary key, preventing duplication in patient information.
  - **Name:** The patient's name.
  - **DOB:** The patient's date of birth.
  - **Gender:** The patient's gender, represented as 0 for female and 1 for male.

123 patient_id	ABC name	dob	123 gender
1,707,243,486	LinhToken	1999-12-12	1
1,706,175,893	update	1973-12-04	0
1,704,823,284	Bill	2003-12-04	1
1,704,267,307	Primary	1970-12-04	1

**Figure 19.** Patients table.

- **Signals:** This table contains metadata of patients and signals by matching patient IDs with their corresponding signals. It includes the following columns:
  - **Signal ID:** A unique identifier for each signal, defined by the Unix timestamp at the moment the signal is measured in ESP32. This serves as the primary key, ensuring that each signal only matches one patient.
  - **Patient ID:** A unique identifier for each patient, identical to the one in the Patients table.
  - **Status:** A status indicator, represented as 0 for deactivated and 1 for activated. This helps the data pipeline find the correct patient to push signal information to match with that patient.

123 signal_id	123 patient_id	123 status
1,707,308,968	1,704,267,307	1
1,707,310,108	1,704,267,307	1
1,707,310,092	1,704,267,307	0
1,708,006,743	1,704,267,307	0
1,708,008,443	1,704,267,307	0
1,708,008,464	1,704,267,307	0
1,708,008,650	1,704,267,307	0
1,708,008,669	1,704,267,307	0
1,708,008,689	1,704,267,307	0
1,708,008,857	1,704,267,307	0
1,708,009,146	1,704,267,307	0

Figure 20. Signals table.

- Time Series Data: This table contains all information about the signal, including:
  - Signal ID: A unique identifier for each signal, identical to the one in the Signals table.
  - Volume: A list of paired values of time and volume that have been processed.
  - Flow: A list of paired values of time and flow rate that have been processed.
  - Median Volume, Voided Volume, Maximum Flow Rate, Average Flow Rate, Time to Maximum Flow Rate, and Flow Time.
  - Comment: The doctor's comment for each signal.

123 signal_id	123 volume	123 flow	123 median_volume	123 voided_volume	123 max_flow_rate	123 avg_flow_rate	123 time_to_max_flow_rate	123 flow_time	123 comment
1,498,255,288	0.0,1.66 0.011,1.69 0.022,1.66 0.011,1.63 0.022,1.66	0.0,0.622 0.011,0.636 0.022,0.622	200.87	197.99197	21.25	4.155915493	5.368	48.191	No Comment
1,498,344,071	0.0,3.0 0.011,3.08 0.022,3.0 0.011,3.07 0.022,3.0	0.0,7.64 0.011,7.67 0.022,7.64	188.68	186.9758	16.38	1.8780024307	5.577	100.111	No Comment
1,707,308,968	0.0,0.57 0.011,0.58 0.022,0.57 0.011,0.57 0.022,0.57	0.0,0.09 0.011,0.07 0.022,0.09	0.62	0.01133	0.09	0.0029261364	0	4.422	No Comment
1,707,310,108	0.0,-1201.14 0.012,-1199.0 0.011,1191.3	-15.18	972.43584	1,726.48	5.1484320203	0.264	189.48	No Comment	
1,708,010,697	0.0,-435.8 0.011,-435.55 0.0353 0.011,3.09 0.022,-433.41	0.0,3.53 0.011,3.09 0.022,-433.41	-433.41	0.88924	3.53	0.1712711864	0	5.742	No Comment
1,708,013,621	0.0,-1.17 0.011,-1.15 0.02,0.14 0.011,14.67 0.02,443.7	0.0,14.04 0.011,14.67 0.02,443.7	443.7	447.3205	50.04	11.384518477	12.496	39.842	No Comment
1,707,310,092	0.0,-166.67 0.011,-166.67 0.0051 0.011,0.51 0.022,-166.39	0.0,0.51 0.011,0.51 0.022,-166.39	-166.39	0.08855	0.51	0.0207474227	0	4.818	test deploy cmt
1,708,006,743	0.0,-1242.87 0.011,-1242.0 0.001 0.011,0.0 0.022,0.0	0.0,0.0 0.011,0.0 0.022,0.0	-1,242.87	0	0	0	0	5.786	No Comment
1,708,008,443	0.0,-0.25 0.011,-0.25 0.02,0.058 0.011,0.6 0.022,0.18	0.0,0.58 0.011,0.6 0.022,0.18	0.18	0.30899	0.69	0.0793502825	0.132	4.444	No Comment
1,708,008,464	0.0,-450.55 0.011,-450.43 0.0125 0.011,1.04 0.022,-449.73	0.0,1.25 0.011,1.04 0.022,-449.73	-449.73	0.22022	1.25	0.0489486553	0	5.049	No Comment
1,708,008,650	0.0,-0.65 0.011,-0.65 0.02,0.144 0.011,1.49 0.022,0.6	0.0,1.44 0.011,1.49 0.022,0.6	0.6	0.97889	1.89	0.2373066667	0.176	4.675	No Comment
1,708,008,669	0.0,0.12 0.035,0.12 0.07,0.0 0.001 0.035,0.02 0.07,0.23	0.0,0.01 0.035,0.02 0.07,0.23	0.23	0.10605	0.06	0.0083013699	0.84	14.525	No Comment
1,708,008,689	0.0,0.99 0.036,1.02 0.072,0.046 0.036,0.44 0.072,1.88	0.0,0.46 0.036,0.44 0.072,1.88	1.88	0.29376	0.46	0.022173913	0	15.048	No Comment
1,708,008,857	0.0,1.59 0.011,1.59 0.022,0.0 0.011,0.02 0.022,1.62	0.0,0.0 0.011,0.02 0.022,1.62	1.62	0.0297	0.05	0.0075842697	0.231	4.466	No Comment
1,708,009,146	0.0,-483.29 0.011,-481.0 0.05233 0.011,48.2 0.022,-454.29	0.0,0.5233 0.011,48.2 0.022,-454.29	-454.29	4.72307	52.33	1.073425	0	4.95	No Comment
1,708,010,356	0.0,-452.42 0.012,-451.98 0.041532 0.012,431.37 0	0.0,0.41532 0.012,431.37 0	-360.895	101.0142	621.77	561.19	0.18	0.78	No Comment
1,708,010,389	0.0,4.45 0.011,4.46 0.022,0.0002 0.011,0.0 0.022,4.46	0.0,0.02 0.011,0.0 0.022,4.46	4.46	0.00022	0.02	0.0000417537	0	5.819	No Comment
1,708,010,461	0.0,0.06 0.011,0.07 0.022,0.238 0.011,2.44 0.022,2	0.0,2.38 0.011,2.44 0.022,2	2	1.4751	3	0.3089861751	0.154	5.324	No Comment
1,708,037,852	0.0,0.59 0.011,0.61 0.022,0.174 0.011,18.22 0.022,322.47	0.0,0.174 0.011,18.22 0.022,322.47	322.47	440.09515	72.89	25.1785084959	0.66	18.029	No Comment
1,708,038,719	0.0,-0.02 0.011,-0.02 0.02,0.205 0.011,2.13 0.022,400.19	0.0,0.205 0.011,2.13 0.022,400.19	400.19	401.33291	110.8	14.4037939203	3.894	28.413	No Comment
1,708,040,421	0.0,0.02 0.011,0.04 0.022,0.1482 0.011,15.42 0.02,174.63	0.0,14.82 0.011,15.42 0.02,174.63	174.63	415.06927	123.09	13.0340483592	27.984	32.395	No Comment
1,708,117,106	0.0,-1.02 0.011,-0.99 0.02,0.2998 0.011,31.2 0.022,180.82	0.0,0.2998 0.011,31.2 0.022,180.82	180.82	177.27468	88.55	11.9465381764	1.265	15.389	No Comment

Figure 21. Time series signal table.

This design ensures a clear and concise representation of the data, facilitating efficient and accurate data retrieval and analysis. It also adheres to the principles of relational database design, promoting data integrity and consistency.

### c. Web application

In the context of web application development, two primary components are integral to the system's operation: the backend and the frontend.

The backend, often referred to as the server-side, is responsible for the core functionalities of the system. It operates behind the scenes, unseen by the end user. Its primary tasks include retrieving information from the database and transmitting it for visualization. Additionally, it receives data from the end user, which it uses to perform CRUD (Create, Read, Update, Delete) operations on the database. In the context of this project, these operations would involve the specific data and signals relevant to the web application.

On the other hand, the frontend, which encompasses the User Interface (UI) and User Experience (UX), serves the purpose of visualizing the data retrieved from the backend. It also provides an interface for users to input data, which is then processed by the backend.

In contemporary web development, a multitude of programming languages are available for both backend and frontend development. These languages, each with their unique features and advantages, will be elaborated upon in **Tables 6** and **Table 7**.

**Table 7.** Decision matrix for backend programming languages.

Criteria	Python	Java	Node.js	Ruby
<b>Complexity</b>	Easy	Moderate	Moderate	Moderate
<b>Performance</b>	Good	Excellent	Good	Good
<b>Database Support</b>	Excellent	Excellent	Good	Good
<b>Host in lambda function</b>	Yes	No	Yes	No

Based on the decision matrix in **Table 6**, Python emerges as the most suitable programming language for this project. Python's simplicity makes it an accessible language for developers, reducing the learning curve and facilitating more efficient code development. While its performance is rated as Good compared to Excellent of Java, it is important to note that for many web applications, the difference in performance between languages is often negligible. Python's database support is excellent, which is crucial for applications that rely heavily on database interactions. Furthermore, Python's compatibility with AWS Lambda functions provides the flexibility to develop serverless applications, a feature that is not available with Java or Ruby. Therefore, considering these factors, Python stands out as the most appropriate choice for this project.

**Table 8.** Decision matrix for frontend programming languages.

Criteria	Nuxt.js	React.js	Angular.js	Vue.js
<b>Learning Curve</b>	Easy	Moderate	Steep	Moderate
<b>Performance</b>	High	High	High	High
<b>Server-Side Rendering (SSR)</b>	Built-in	Requires Next.js	Requires Angular Universal	Requires manual setup
<b>Community Support</b>	Large and growing	Very large	Very large	Large
<b>Flexibility</b>	High (Vue.js + additional features)	High	Moderate (strict guidelines)	High

Based on the decision matrix of **Table 7**, Nuxt.js emerges as a highly suitable choice for front-end development. Its ease of learning makes it accessible to developers of varying skill levels. It offers

high performance, which is a critical factor for ensuring a smooth and responsive user experience. One of the standout features of Nuxt.js is its built-in server-side rendering (SSR), which can significantly improve the performance and SEO of web application. Additionally, Nuxt.js has a large and growing community, which can be a valuable resource for troubleshooting and learning. Lastly, Nuxt.js offers high flexibility, thanks to its foundation on Vue.js and additional features. These combined attributes make Nuxt.js a compelling choice for front-end development, potentially outshining other frameworks like React.js, Angular.js, and Vue.js in certain scenarios.

To ensure the optimal functioning of the application, which in this case involves performing CRUD operations for a PostgreSQL database and visualizing the retrieved data for the end user, the backend will be structured in the form of an API. The frontend will employ logic to call the API based on user interaction. As mentioned in Section B, Lambda functions perform well in ETL tasks. In this context, the backend CRUD operations can be viewed as small ETL tasks. Therefore, the backend is written using Python with Fast API as the core library, and the APIs are hosted using Lambda functions. The APIs needed for the backend include:

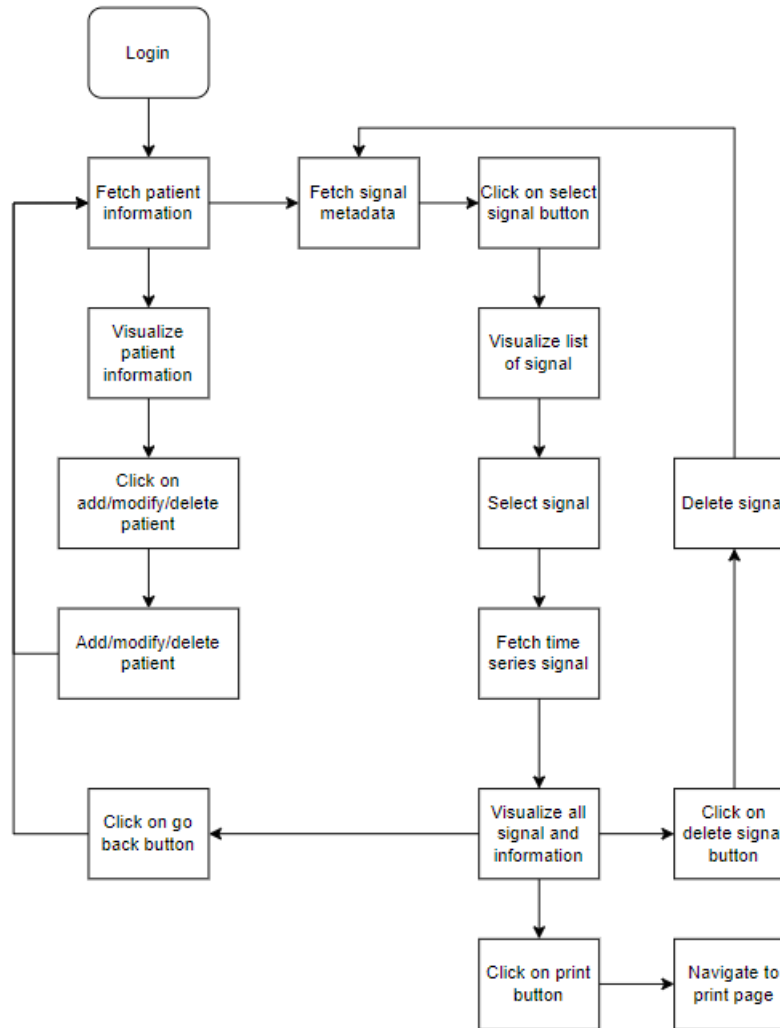
- GET: Used to retrieve information from the PostgreSQL database, equivalent to ‘Read’ in CRUD.
  - Fetch patient info: No input required. The output includes basic patient information such as ID, name, date of birth, and gender.
  - Fetch signal metadata: The input is the Patient ID. The output includes the Patient ID, Signal ID, and Status.
  - Fetch time series: The input is the Signal ID. The output includes all information about the signal, encompassing all columns of the Time Series Signal table.
- POST: Used to add information to the PostgreSQL database, equivalent to ‘Create’ in CRUD.
  - Add patient: The input includes the name, date of birth, and gender of the patient. The output will be pushed into the database, where the Patient ID is auto-generated based on the Unix time at the moment the user creates the patient.
- PUT: Used to update information in the PostgreSQL database, equivalent to ‘Update’ in CRUD.
  - Modify Patient: The input includes the patient ID, name, date of birth, and gender of the patient. The output will be the new info of the patient with the corresponding patient ID.
  - Activate Patient: The input is the patient ID. The output will change the Status of the corresponding patient ID to 1. This API acts as a marker for the data pipeline to push data into the correct patient.
  - Modify comment: The input is the signal ID and the user’s comment. The output will be the new comment for the corresponding signal ID.

- **DELETE:** Used to delete information from the PostgreSQL database, equivalent to ‘Delete’ in CRUD.
  - Delete Patient: Deletes all information of a Patient, including their corresponding signal. The input is the patient ID.
  - Delete Signal: Deletes all information about a signal. The input is the signal ID.

GET	/patient/get	Fetch Patient Info	▼
GET	/signal_meta/{patient_id}	Fetch Signal Metadata	▼
GET	/signal_info/{signal_id}	Fetch Times Series	▼
POST	/patient/add	Add Patient	▼
PUT	/patient/modify/{patient_id}	Modify Patient	▼
DELETE	/patient/delete/{patient_id}	Delete Patient	▼
PUT	/signal/activate/{patient_id}	Activate Signals	▼
DELETE	/signal/delete/{signal_id}	Delete Signals	▼
PUT	/signal/comment/{signal_id}	Modify Comment	▼

**Figure 22.** FAST API for backend.

With the FAST API listed above, the flowchart of web application is as follows:



**Figure 23.** Flow chart of web application.

Upon user login, the system initiates a GET API call to fetch patient information. This information is subsequently visualized on the main page of the web application. Concurrently, the patient ID, obtained from the fetched information, is utilized to retrieve signal metadata.

The user interface provides options to add, modify, or delete patient information. Upon clicking the corresponding button, an API call is triggered to perform the requisite CRUD operation. Following this, the patient information is fetched anew to reflect any updates.

The signal metadata, once fetched, enables the visualization of a list of signals upon the user clicking the ‘Select Signal’ button. If a user selects a specific signal, the system fetches the time series signal based on the chosen signal ID. This information is then visualized in both chart and text forms.

In the event of the user clicking the ‘Delete’ button, the selected signal is deleted, and the interface reverts to the main page. The signal metadata is fetched again to ensure up-to-date information is displayed. If the ‘Print’ button is clicked, a print window appears, allowing the user to select their



desired settings before initiating the print command.

Finally, if the user clicks the ‘Go Back’ button, the system navigates back to the main page and fetches the patient information again, ensuring the user is always presented with the most current data. This systematic and efficient process ensures a seamless user experience while maintaining the integrity and accuracy of the data.

## D. Final Prototype



Figure 24. TeleUroflow Login Interface

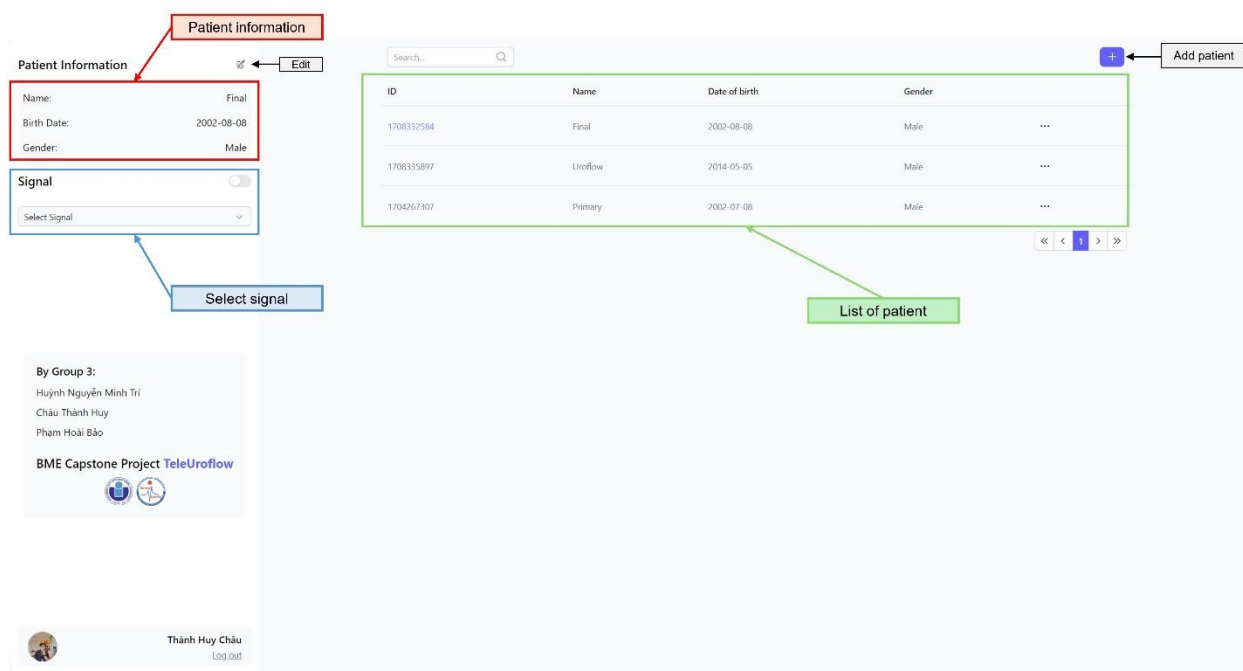


Figure 25. TeleUroflow patient information dashboard

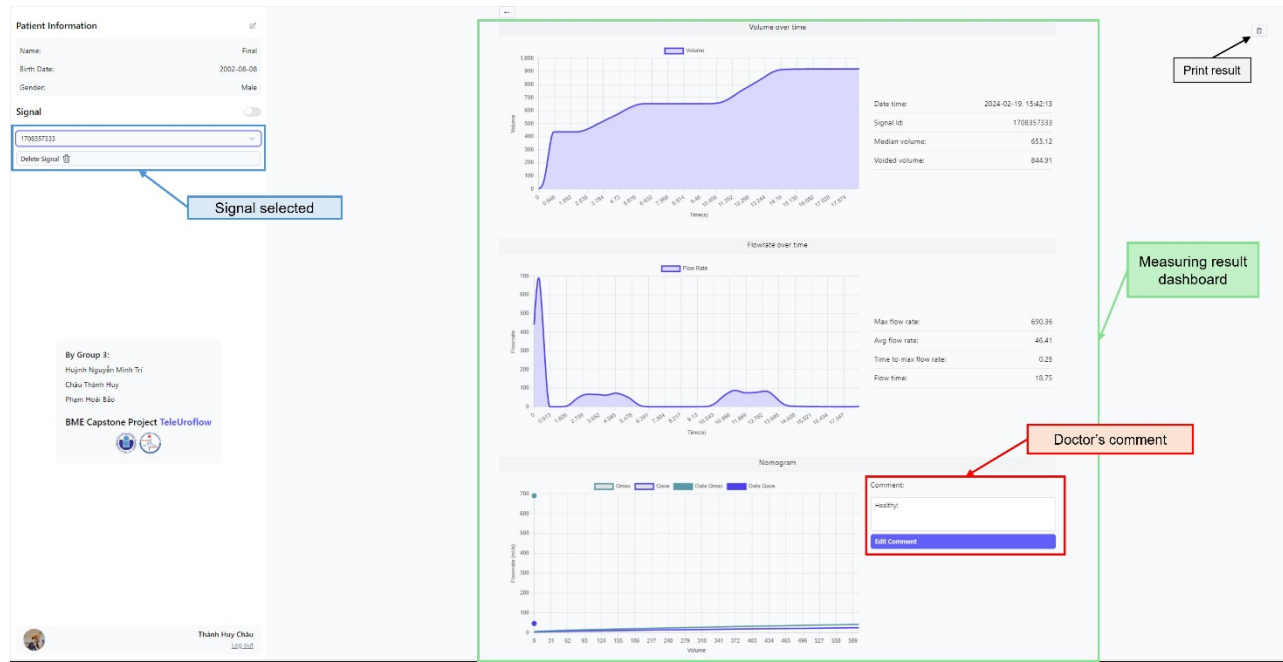


Figure 26. TeleUroflow measuring dashboard

The web application operates via a structured process: initially, doctors log in to their accounts through the login interface (Figure 24). Subsequently, the system navigates to the patient information dashboard (Figure 25), where doctors have several options, including selecting an existing patient from the list (list of patients), adding a new patient (+), choosing a signal type (select signal), updating patient information (Edit), and toggling the recording of new signals from the uroflowmeter (switch next to Signal). Following the recording of the doctor's actions, the system transitions to the corresponding section. Post-measurement and signal type selection, the interface proceeds to the dashboard that exhibits the measurement results (Figure 26). This dashboard showcases three principal chart types:

- **Volume Over Time Graph:** This graph monitors the cumulative volume of urine excreted throughout the urination process. It is instrumental in visualizing the quantity of urine released as time elapses and is valuable for pinpointing any anomalies in urine output.
- **Flow Rate Over Time Graph:** This graph delineates the urine flow rate in real-time, from the commencement to the conclusion of urination. It is essential for identifying the peak flow rate, the swiftness with which it is attained, variations in flow during urination, and the overall flow duration. Additionally, it aids in detecting patterns that may suggest conditions such as an obstruction or overactive bladder.
- **Nomogram Graph:** In uroflowmetry, the nomogram is a tool used to juxtapose the patient's flow rate and voided volume with standard values. It serves as a diagnostic instrument to categorize urinary flow as either normal or indicative of an obstruction or other complications.

### III. Testing and Performance Evaluation

#### A. Testing plans

Because most of our tasks were related to designing and interfacing the Uroflowmeter with the web, the testing plan included functional, database development and signal transmission tests. The detailed plan is given in **Appendix 1**. To enhance the signal quality and evaluate transmission time, we tested the latency in database update and the performance of our signal processing strategies.

##### a. Database update latency

In terms of database update latency, it is observed that the latency increases with the volume of data, implying that larger volumes require longer upload times. Given that the bladder's capacity for storing urine ranges from 350ml to 650ml, we will conduct two tests: one with the minimum volume and one with the maximum volume. These tests will measure the time taken to upload the data. Subsequently, we will compute the average of the received times to estimate the maximum and minimum upload durations.

##### b. Signal processing

In real-world data recording, fluctuations and errors are unavoidable due to the characteristics of load cells, which measure weight changes in milliseconds. Consequently, despite an overall increasing trend, the data exhibits significant fluctuations when examined closely, leading to alternating positive and negative flow rates. These negative flow rates, however, should not exist and are merely errors. Therefore, signal processing is essential to correct these anomalies. The comparison of volume over time and flow rate over time, both before and after applying a moving average to the weight, underscores the critical role of signal processing in this project.

##### c. Functionality check

Finally, all functions listed above such as adding patient, modifying patient, deleting patient will be checked for both visualization and interaction. This part includes using each function, API, and check whether it visualizes correctly after interacting with the web application.

#### B. Performance Evaluation

##### a. Database update latency

The test results are presented in **Table 9**. This approach ensures a comprehensive understanding of the system's performance across a range of volumes.

- The minimum time uploaded estimated are:

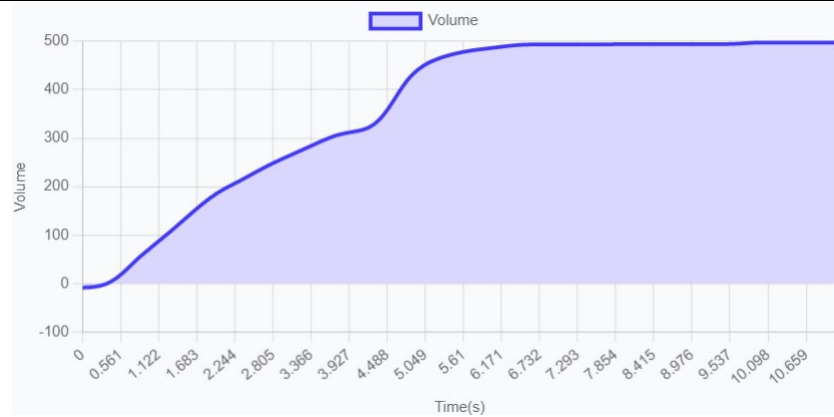
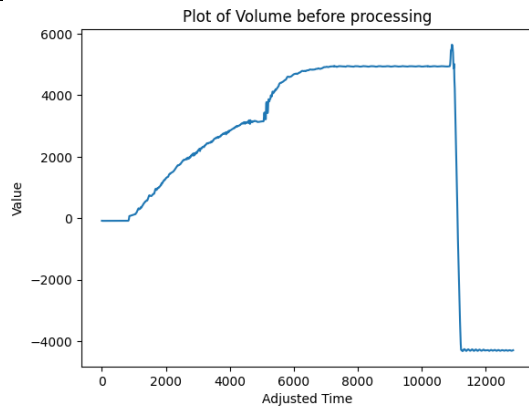
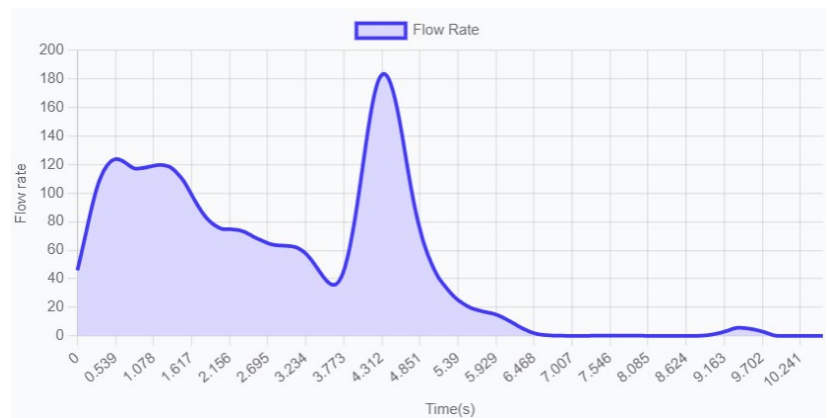
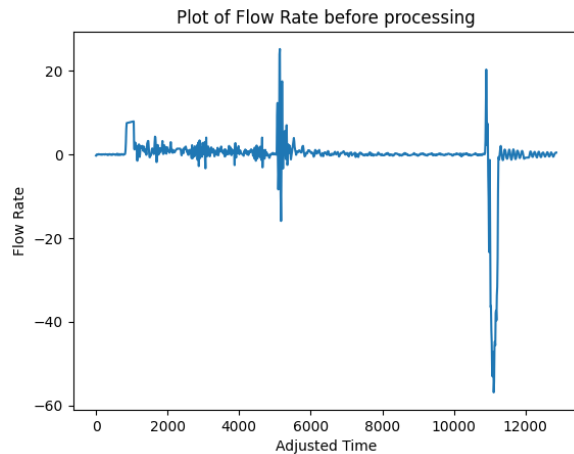
$$\frac{0.367 + 0.385 + 0.332}{3} = 0.361 \text{ (second)}$$

- The maximum time uploaded estimated are:

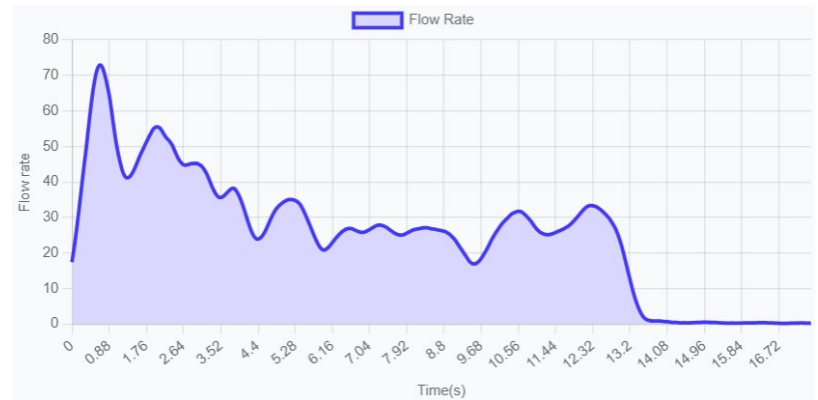
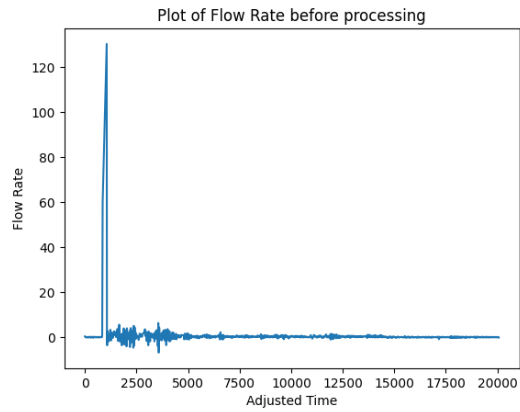
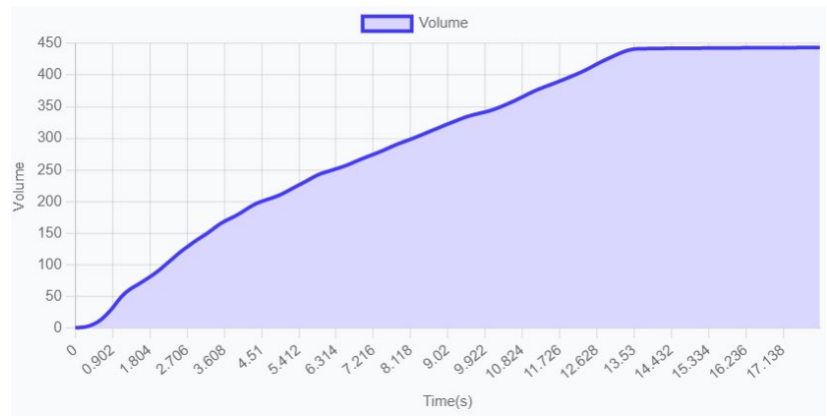
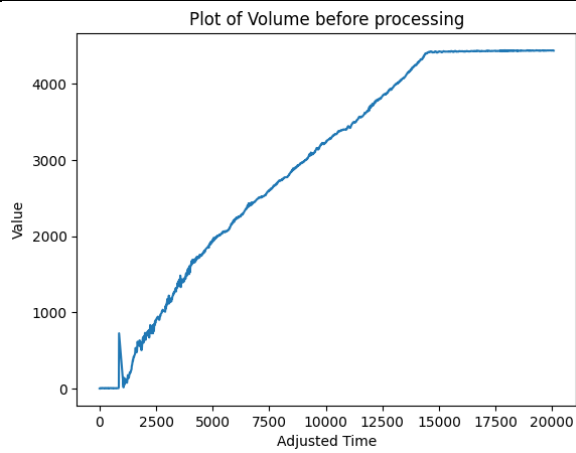
$$\frac{0.680 + 0.623 + 0.637}{3} = 0.646 \text{ (second)}$$

**Table 9.** Database update latency trial.

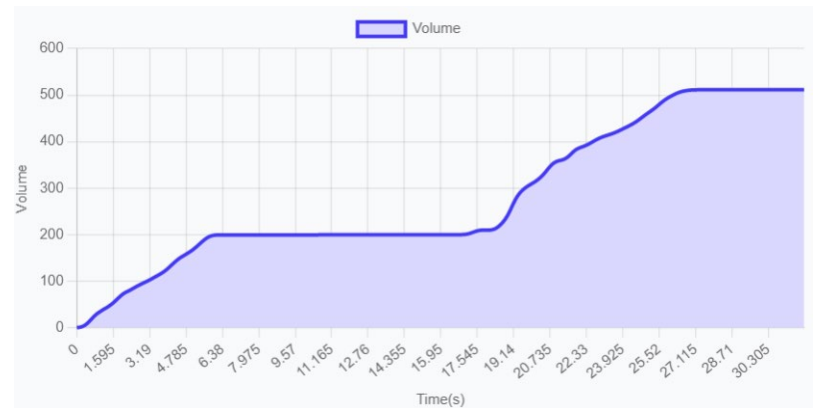
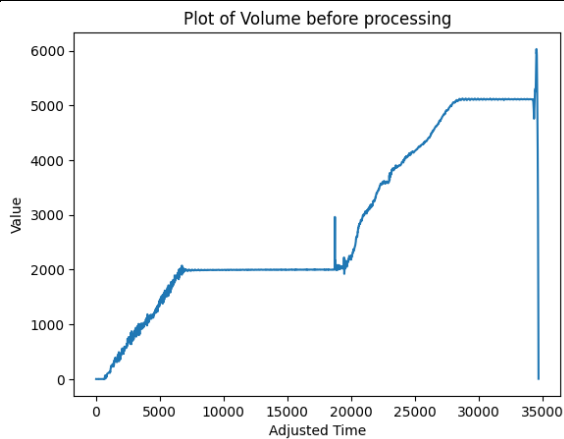
<b>Volume</b>	<b>350ml</b>	<b>650ml</b>
<b>1<sup>st</sup> Trial</b>	0.367s	0.680s
<b>2<sup>nd</sup> Trial</b>	0.385s	0.623s
<b>3<sup>rd</sup> Trial</b>	0.332s	0.637s

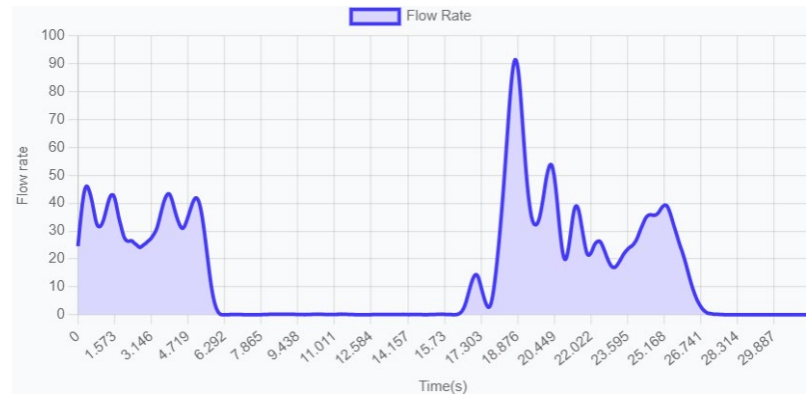
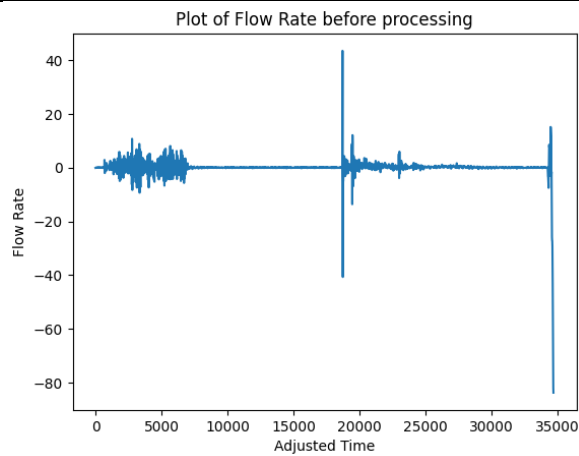
**b. Signal processing****Table 10.** Comparison of signal before and after processing.**Trial****Before processing****After processing****1st**

2nd



3rd





### c. Functionality check

**Patient Information**

Name: final

Birth Date: 2002-08-08

Gender: Male

Signal ☐

Select Signal

**By Group 3:**

Huỳnh Nguyễn Minh Trí

Châu Thành Huy

Phạm Hoài Bảo

**BME Capstone Project**

**TeleUroflow**

Trí Huỳnh

Log out

Search...

ID	Name	Date of birth	Gender
1707243486	LinhToken	1999-12-12	Male
1706175893	update	1973-12-04	Female
1704823284	Bill	2003-12-04	Male
1704267307	Primary	1970-12-04	Male
1708180124	string	2003-12-04	Female
1708332584	final	2002-08-08	Male

« < 1 > »

Figure 27. Fetch patient information.

**Signal** ☐

Select Signal

Select Signal

1708037852

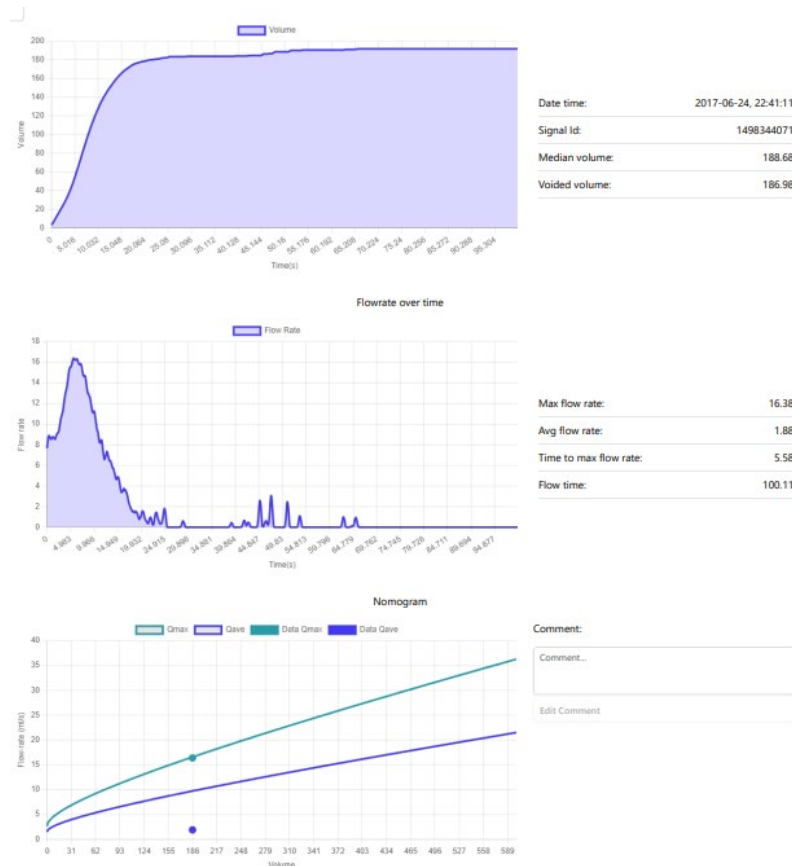
1708038719

1708040421

1708117498

1708117893

Figure 28. Fetch signal metadata.



**Figure 29.** Fetch time series signal.

Patient Name

Date Of Birth

Gender

**Add Patient**

a)

Patient Name

Date Of Birth

Gender

**Edit Patient**

b)

**Figure 30.** a) Add patient b) Modify patient.

Male ...

Edit

Delete

a)

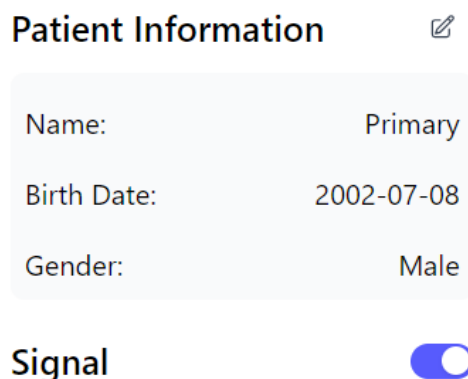
Do You Want To Delete This Patient ?


**Yes**

**No**


b)

**Figure 31.** a) Delete drop down box b) Delete pop up window.

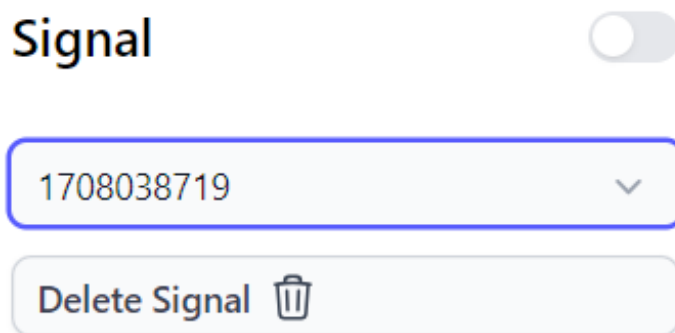



**Patient Information** 


Name:	Primary
Birth Date:	2002-07-08
Gender:	Male


**Signal** 

**Figure 32.** Activate signal.



**Signal** 

1708038719 

Delete Signal 

**Figure 33:** Delete signal.

#### d. Discussion

As delineated in Section A, the results of our experiment have met our initial expectations. The machinery, despite its inherent limitations, has performed optimally, capturing the signals as anticipated.

The subsequent signal processing phase has exceeded our expectations, as evidenced by the data presented in **Table 10**. The processing algorithm has been successful in eliminating points where the volume decreases, a phenomenon caused by the alternating positive and negative flow rates. Furthermore, it has effectively excluded the superfluous data recorded when the container was removed from the machine, a fact clearly demonstrated in the first trial presented in **Table 10**.

Despite the intensive signal processing, the pipeline has maintained an efficient upload speed, as shown in **Table 9**. Although the speed is significantly dependent on the size of the data, the data volume does not exceed 650ml, which is the maximum capacity of a normal bladder. This ensures that the maximum time required to upload the data to the database remains under one minute.

Finally, as demonstrated in the functionality check section, all functions and APIs are operating efficiently. The Create, Read, Update, and Delete (CRUD) operations are performing as expected, and the desired signals have been visualized clearly. This comprehensive analysis affirms the robustness of our system and its potential for further deployment.



## C. Product Economic Evaluation

Regarding the cost of our urinary flow measurement system, it's a blend of inherited hardware and operational software costs. The hardware component was passed down to us by a previous Capstone team, which means we incurred no expense in this area. Nonetheless, to provide a clear cost evaluation and comparison, we've estimated the cost of each hardware component as listed in **Table 12**.

**Table 11.** Price list of main components on the circuit.

Number	Components	Cost
1	ESP32 NodeMCU LuaNode32 38 Pin Wifi Transceiver Module	170,000VND
2	ADC Converter Circuit 24bit Loadcell HX711	16,000VND
3	RGB Led Driver Circuit	2,800VND
4	Real Time Circuit RTC DS1302	79,000VND
5	SD card breakout board	12,000VND
6	18650 Battery Charger Circuit 1 Battery 1S With Protection 1A	26,000VND
<b>Total cost</b>		<b>305,800VND</b>

On the software front, there are two main types of expenses: hosting and ETL (Extract, Transform, and Load). To simplify our calculations, we assume our system operates within the University of Medicine and Pharmacy Hospital, Ho Chi Minh City, catering to approximately 6,500 patients daily. Assuming about 25% of these patients, equating to roughly 1,650 individuals, require urinary flow measurements using our device.

Hosting costs are predicated on a database supporting information for 49,500 patients monthly (assuming mainly new patients each day, with repeat patients being negligible). Based on AWS's pricing calculator, the estimated monthly hosting cost is around \$1,4. However, this cost is expected to rise with an increase in patient numbers.

8.20 GB per month x 0.138 USD x 1 instances = 1.1316 USD (Storage Cost)  
**Storage pricing (monthly): 1.13 USD**

**Figure 34.** Hosting pricing calculation performed by AWS cost analysis software.

For ETL costs (**Figure 35**), our calculations conclude that ETL services will be free for up to 2,200 patients daily. Beyond this threshold, ETL costs will apply at a rate of \$0.01 per additional 100 patients. This pricing model ensures that the system remains cost-effective for smaller patient loads while scaling modestly with increased usage.

Amount of memory allocated:  $128 \text{ MB} \times 0.0009765625 \text{ GB in a MB} = 0.125 \text{ GB}$   
Amount of ephemeral storage allocated:  $512 \text{ MB} \times 0.0009765625 \text{ GB in a MB} = 0.5 \text{ GB}$

Pricing calculations

$35,200 \text{ requests} \times 125 \text{ ms} \times 0.001 \text{ ms to sec conversion factor} = 4,400.00 \text{ total compute (seconds)}$   
 $0.125 \text{ GB} \times 4,400.00 \text{ seconds} = 550.00 \text{ total compute (GB-s)}$   
 $550.00 \text{ GB-s} - 400000 \text{ free tier GB-s} = -399,450.00 \text{ GB-s}$   
 $\text{Max} (-399450.00 \text{ GB-s}, 0) = 0.00 \text{ total billable GB-s}$   
Tiered price for: 0.00 GB-s  
Total tier cost = 0.00 USD (monthly compute charges)

$35,200 \text{ requests} - 1000000 \text{ free tier requests} = -964,800 \text{ monthly billable requests}$   
 $\text{Max} (-964800 \text{ monthly billable requests}, 0) = 0.00 \text{ total monthly billable requests}$   
 $0.50 \text{ GB} - 0.5 \text{ GB (no additional charge)} = 0.00 \text{ GB billable ephemeral storage per function}$


**Lambda costs - With Free Tier (monthly): 0.00 USD**

**Figure 35.** ETL pricing calculation performed by AWS cost analysis software.

## IV. Project Management

**Table 12.** Project timeline and tasks.

No.	Tasks	Week														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Choose initial idea, literature review															
2	Research and compare different database															
3	Prepare the working plan															
4	Receive the Uroflowmeter and connect ESP32 with the computer to collect data from the Uroflowmeter															
5	Investigate the data type and format collected from the Uroflowmeter															
6	Learn to use and access the database															
7	Upload raw data into intermediate storage for temporary preprocessing															
8	Transform data into compatible format for the database															
9	Upload processed data into the final storage															
10	Learn web app development															
11	Develop web-based app to visualize and analyze data															
12	Interface the web app with the database															
13	Tests and evaluate the entire system															
14	Project presentation															

 Planned time

 Flow time

**Table 13.** Member and task divisions.

No.	Member	Tasks	Evaluation
1	Pham Hoai Bao	Project planning: define project scope and objectives, identify requirements, and manage the project timeline and milestones Manage the project logbook, report Configure web app code, web app interface RESTful API for data communication Handle data storage and retrieval	<b>Complete</b>
2	Chau Thanh Huy	Set up ESP32 for data collection Establish communication protocol with uroflowmeter Investigate data format and requirements for biomedical data security in database Research and design user interface for the web application	<b>Complete</b>
3	Huynh Nguyen Minh Tri	Design PostgreSQL database schema and pipeline for uroflowmetry data Implement database connectivity Prepare the application for deployment and set up hosting environment Back-end Development: Research and implement, database server and connection with the ESP32 workflow	<b>Complete</b>

Our group consists of individuals from three distinct sections within the biomedical engineering department, making it challenging to find a mutual free period due to varying schedules, assignments, or personal tasks. However, with the significant decline in the Covid-19 epidemic, we are now able to attend offline classes and labs, enabling us to gather at the university. We utilize breaks and post-class time for project discussions, which is especially useful for clarifying queries that are hard to explain via social media.

While developing the telemedicine platform, we faced initial hurdles in familiarizing ourselves with the system and had to revise it multiple times to pinpoint essential features and subpages. The user interface also needed enhancement to ensure it was visually appealing, user-friendly, and functional. Thanks to Mr. Linh's invaluable guidance, we managed to launch an initial version of the platform that meets key requirements. His advice not only helped us improve the platform's aesthetics but also inspired us to design a meaningful brand logo that encapsulates the essence of our medical project.

## V. Conclusion and Discussion

In conclusion, our team has successfully integrated IoT into our previously developed Uroflowmeter, resulting in the creation of a web-based system for the remote storage and visualization of Uroflowmetry data. Uroflowmetry (UF) is currently considered the standard for assessing men with benign prostatic obstruction (BPO) and proven to be a valuable tool for decision-making in the treatment of individuals experiencing lower urinary tract symptoms (LUTS). The increasing adoption of telemedicine and telehealth in recent years has offered cost-effective treatment alternatives for both patients and healthcare providers. Through the integration of IoT with the Uroflowmeter, patients can conveniently perform tests at home, eliminating the need to visit central hospitals. Utilizing the AWS S3 storage and PostgreSQL database engine, we interface the Uroflowmeter with our web-based system, allowing clinicians to remotely access and manage UF data from patients on a daily basis. This capability could, in turn, alleviate the burden on the healthcare sector.

We also applied weighted moving average smoothing technique to preprocess raw signal collected from the sensor. In brief, the signal processing step had successfully eliminated points of volume decrease caused by alternating positive and negative flow rates. The processing algorithm also filtered out extraneous data recorded during container removal, ensuring accuracy and integrity of data. Despite the intensive signal processing, the pipeline maintains efficient upload speeds. This guarantees that the data upload to the database remains under one minute, providing a streamlined and effective process.

On the other hand, the system presented several limitations that need to be addressed in future study. One notable limitation is the potential oversight of activating the patient button. This button is crucial for associating the recorded signal with the respective patient during the measurement process. If a user forgets to activate the button, the pipeline may not correctly attribute the signal to the intended individual, leading to inaccuracies in data analysis. In practical situations, forgetting to activate the button results in immediate data transmission, and currently, there is no established solution to address this issue. Additionally, tasks such as going to the bathroom may be challenging as the device lacks the capability to handle interruptions and resume measurements, posing a practical constraint. While a backup solution is in place with the S3 storage containing raw data, future developments may involve incorporating functions in the app to manage and process data that gets delayed due to overlooked signal activation.

Another aspect for potential improvement lies in the sensitivity of the sensor. The sensor's high sensitivity may cause it to record data continuously in unstable environments. This sensitivity can lead to the transmission of erroneous data to the system, affecting the overall reliability of the device. Although a solution is currently implemented through a data deletion button, the system may face challenges if instability persists. To enhance the device's robustness, future developments could focus on refining the sensor's sensitivity to minimize unnecessary data recording in unstable conditions. This would contribute to the overall effectiveness and dependability of the system, ensuring that the recorded data accurately reflects the patient's actual physiological state.

## **VI. Acknowledgment**

First and foremost, we extend our heartfelt thanks to Professor Vo Van Toi for the invaluable knowledge he imparted on engineering advancements, devices, methods, various techniques, and the skills necessary for executing this project. His generosity, patience, and insightful advice were instrumental in navigating the challenges encountered during the project and drove us to achieve our utmost. We are equally thankful to Mr. Nguyen Le Y for his detailed guidance on this project. He provided clear directions and essential tasks, and his enthusiasm served as a significant source of inspiration, pushing us to successfully complete the project. His expertise helped demystify complex issues, ensuring a smoother journey in our academic endeavors.

Secondly, we want to express our gratitude toward Dr. Tran Le Giang, our lecturer in Biosignal Processing and Internet of Things for Healthcare classes. The knowledge and skills he taught us had been instrumental during the entire period of the project, especially during building the idea and requirements for the web.

Our appreciation also goes to our Teaching Assistants, Ms. Le Thi Thuy Tien and Mr. Vo Minh Thien, who were with us every step of the way, enriching our project with creative ideas and constructive feedback for system design improvements.

Special acknowledgment is due to Mr. Le Hoang Linh for his meticulous feedback and advice on developing web-based software throughout the project. His support and problem-solving expertise were crucial to the success of our tele-Uroflowmetry system and platform.

Finally, we express our profound gratitude to the School of Biomedical Engineering at the International University for its steadfast support. The school's encouragement has not only afforded us additional opportunities for innovation but also inspired us to delve deeper into the realms of science and technology, unlocking our creative potential.

## VII. References

- [1] M. H. Kristensen, H. Elfeki, S. Sinimäki, S. Laurberg, and K. J. Emmertsen, “Urinary dysfunction after colorectal cancer treatment and impact on quality of life—a national cross-sectional study in males,” *Colorectal Dis.*, vol. 23, no. 2, pp. 394–404, 2021, doi: 10.1111/codi.15554.
- [2] Y. Aoki, H. W. Brown, L. Brubaker, J. N. Cornu, J. O. Daly, and R. Cartwright, “Urinary incontinence in women,” *Nat. Rev. Dis. Primer*, vol. 3, no. 1, Art. no. 1, Jul. 2017, doi: 10.1038/nrdp.2017.42.
- [3] X. Yang, H. Chen, Y. Zheng, S. Qu, H. Wang, and F. Yi, “Disease burden and long-term trends of urinary tract infections: A worldwide report,” *Front. Public Health*, vol. 10, p. 888205, Jul. 2022, doi: 10.3389/fpubh.2022.888205.
- [4] L. Stothers and A. J. Macnab, “Home-based monitoring of lower urinary tract health: simultaneous measures using wearable near infrared spectroscopy and linked wireless scale,” in *Biophotonics in Exercise Science, Sports Medicine, Health Monitoring Technologies, and Wearables III*, SPIE, Mar. 2022, pp. 55–60. doi: 10.1117/12.2609782.
- [5] E. O’Connor, A. Nic an Riogh, M. Karavitakis, S. Monagas, and A. Nambiar, “Diagnosis and Non-Surgical Management of Urinary Incontinence – A Literature Review with Recommendations for Practice,” *Int. J. Gen. Med.*, vol. 14, pp. 4555–4565, Aug. 2021, doi: 10.2147/IJGM.S289314.
- [6] S. Elneil et al., “An International Continence Society (ICS) report on the terminology for female pelvic floor fistulas,” *Neurourol. Urodyn.*, vol. 39, no. 8, pp. 2040–2071, 2020, doi: 10.1002/nau.24508.
- [7] K. Kuoch, “Exploration of Bladder and Bowel Anxieties and the Role of Socio-Cognitive Processes”.
- [8] U. F. O. Themes, “Noninvasive Urodynamics,” *Abdominal Key*. Accessed: Feb. 13, 2024. [Online]. Available: <https://abdominalkey.com/noninvasive-urodynamics/>
- [9] S. D. Pandolfo et al., “A Novel Low-Cost Uroflowmetry for Patient Telemonitoring,” *Int. J. Environ. Res. Public Health*, vol. 20, no. 4, Art. no. 4, Jan. 2023, doi: 10.3390/ijerph20043287.
- [10] D. Amparore et al., “Forecasting the Future of Urology Practice: A Comprehensive Review of the Recommendations by International and European Associations on Priority Procedures During the COVID-19 Pandemic,” *Eur. Urol. Focus*, vol. 6, no. 5, pp. 1032–1048, Sep. 2020, doi: 10.1016/j.euf.2020.05.007.
- [11] V. M. Grippa and S. Kuzmichev, *Learning MySQL*. O’Reilly Media, Inc., 2021.
- [12] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, “MongoDB Vs PostgreSQL: A comparative study on performance aspects,” *GeoInformatica*, vol. 25, no. 2, pp. 243–268, Apr. 2021, doi: 10.1007/s10707-020-00407-w.
- [13] P. Filip and L. Čegan, “Comparison of MySQL and MongoDB with focus on performance,” in *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, Nov. 2020, pp. 184–187. doi: 10.1109/ICIMCIS51567.2020.9354307.
- [14] H. Vera-Olivera, R. Guo, R. C. Huacarpuma, A. P. B. Da Silva, A. M. Mariano, and M. Holanda, “Data Modeling and NoSQL Databases - A Systematic Mapping Review,” *ACM Comput. Surv.*, vol. 54, no. 6, p. 116:1-116:26, Jul. 2021, doi: 10.1145/3457608.
- [15] L. Rothenhäusler, “d3.js and its potential in data visualization,” *Technische Hochschule Brandenburg*,

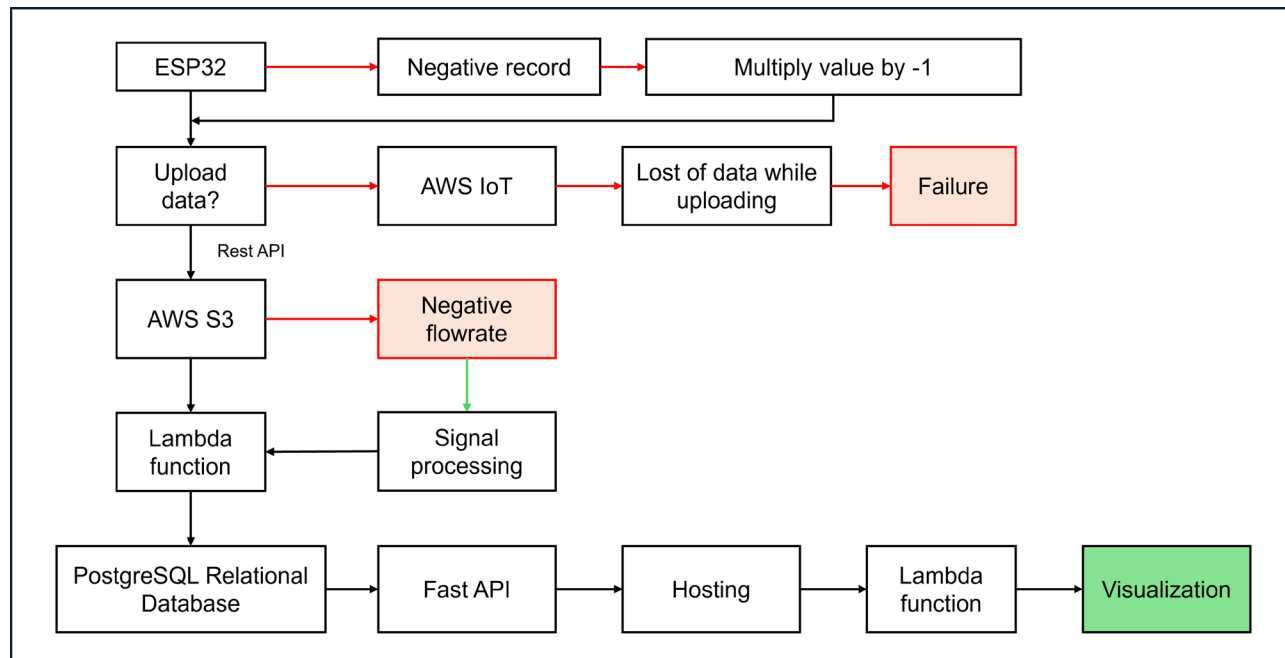
2022. doi: 10.25933/opus4-2853.

- [16] S. Benbba, “Comparison of D3.js and Chart.js as visualisation tools,” 2021, Accessed: Feb. 19, 2024. [Online]. Available: <https://trepo.tuni.fi/handle/10024/131287>
- [17] S. Mishra, “Getting Started with Highcharts,” in *Practical Highcharts with Angular : Your Essential Guide to Creating Real-time Dashboards*, S. Mishra, Ed., Berkeley, CA: Apress, 2023, pp. 1–12. doi: 10.1007/978-1-4842-9181-8\_1.
- [18] U. Rahardja, “Using Highchart to Implement Business Intelligence on Attendance Assessment System based on Yii Framework,” *Int. Trans. Educ. Technol.*, vol. 1, no. 1, pp. 19–28, Nov. 2022, doi: 10.33050/itee.v1i1.176.
- [19] B. Gupta, P. Mittal, and T. Mufti, “A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services,” presented at the *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020*, 27-28 February 2020, Jamia Hamdard, New Delhi, India, Mar. 2021. Accessed: Feb. 20, 2024. [Online]. Available: <https://eudl.eu/doi/10.4108/eai.27-2-2020.2303255>
- [20] A. Wittig and M. Wittig, *Amazon Web Services in Action, Third Edition: An In-depth Guide to AWS*. Simon and Schuster, 2023.
- [21] G. L. Somlo, “Toward a Trustable, Self-Hosting Computer System,” in *2020 IEEE Security and Privacy Workshops (SPW)*, May 2020, pp. 136–143. doi: 10.1109/SPW50608.2020.00039.
- [22] J. Attardi, “Introduction to Netlify CMS,” in *Using Gatsby and Netlify CMS: Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS*, J. Attardi, Ed., Berkeley, CA: Apress, 2020, pp. 1–12. doi: 10.1007/978-1-4842-6297-9\_1.
- [23] J. Attardi, “The Netlify CMS Application,” in *Using Gatsby and Netlify CMS: Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS*, J. Attardi, Ed., Berkeley, CA: Apress, 2020, pp. 57–70. doi: 10.1007/978-1-4842-6297-9\_5.
- [24] H. Abu-Libdeh et al., “Learned Indexes for a Google-scale Disk-based Database.” *arXiv*, Dec. 23, 2020. doi: 10.48550/arXiv.2012.12501.
- [25] M. U. Hassan, I. Yaqoob, S. Zulfiqar, and I. A. Hameed, “A Comprehensive Study of HBase Storage Architecture—A Systematic Literature Review,” *Symmetry*, vol. 13, no. 1, Art. no. 1, Jan. 2021, doi: 10.3390/sym13010109.
- [26] Z. Zhou, J. Pourqasem, and S. Sayadmanesh, “Review of prime issues in big data storage,” *Big Data Comput. Vis.*, vol. 1, no. 4, pp. 191–199, Dec. 2021, doi: 10.22105/bdcv.2022.325253.1040.
- [27] “Bigtable: Fast, Flexible NoSQL,” Google Cloud. Accessed: Feb. 20, 2024. [Online]. Available: <https://cloud.google.com/bigtable>



## VIII. Appendices

### Appendix 1. Testing plan.



### Appendix 2. Source code.

GitHub link: [Uroflow software](#)

Webpage link: [TeleUroflow](#)