

Development of Mobile Application for Smart Flood Warning System

Lee Jia Rock¹, Mohd Fadly Abd Razak^{1*}

¹ Faculty of Electrical Engineering Technology,
Universiti Tun Hussein Onn Malaysia, 84600, Pagoh, Johor, MALAYSIA.

*Corresponding Author: fadly@uthm.edu.my

DOI: <https://doi.org/10.30880/peat.2024.05.01.018>

Article Info

Received: 27 December 2023

Accepted: 17 January 2024

Available online: 15 June 2024

Keywords

Warning System, Flood, Mobile
Application, Real-time Data
Visualization, Alerting Mechanism

Abstract

To overcome the escalating flood events globally, a flood management and early warning systems has become a necessity. This project addresses these challenges by developing a mobile application for Smart Flood Warning System, offering real-time flood data visualization, predictive analytics, and proactive alerting mechanisms. The system is integrated with advanced sensors for continuous data collection on water levels, rainfall, and flow rates, processed via sophisticated algorithms for flood monitoring. A user-centric dashboard and hotspot area on map fragment empowers users to identify flood hotspots and take timely response action to flood events. The project aims to enhance community preparedness by enabling timely alerts via SMS channels to residents in flood-prone areas. Rigorous testing and validation in real-world scenarios ensure system reliability and responsiveness. The anticipated outcomes include an accessible dashboard, an automation warning system, and increased community resilience through informed decision-making. Ultimately, this Smart Flood Warning System represents a pivotal step in fortifying flood management globally, safeguarding lives, and minimizing the impact of recurrent flood events.

1. Introduction

The rise in flood occurrences across various regions has led to an urgent need for robust flood management strategies. In response, a smart flood warning system has been conceptualized, integrating mobile communication to enhance its efficacy. This innovative software application serves as a comprehensive dashboard, collating and processing real-time flood data to provide actionable insights. Its primary objective is to visualize critical flood-related information adaptable to diverse geographical contexts. By leveraging cutting-edge sensors to monitor water levels, rainfall intensity, and flow rates, this system delivers a multifaceted approach to flood management.

At its core, the software offers an intuitive user interface, utilizing geographic information systems (GIS) and mapping functionalities [1]. These features empower users to identify flood-prone areas, predict potential trajectories, and evaluate vulnerabilities with remarkable precision [2]. The application's accessibility across various devices extends its utility to stakeholders ranging from local authorities to emergency response teams and the public. Such inclusivity enhances its effectiveness as a versatile tool for real-time decision-making in flood-prone regions worldwide.

Furthermore, this software fosters collaboration among diverse entities involved in flood management. It facilitates seamless data sharing, timely notifications, and informed decision-making, significantly bolstering collective capabilities to respond proactively to flood events. The system's adaptability allows for automated alerts and warnings, ensuring timely dissemination of crucial information to residents in vulnerable areas [3]. By offering an advanced visualization tool for flood data and hotspot identification, this application signifies a significant leap forward in global flood preparedness. Its capability to empower communities with informed decision-making tools holds promise in safeguarding lives and essential resources against the recurring threat of floods on a global scale.

The inadequacies in current flood warning systems—inefficient data collection, analysis, and timely communication—underscore the urgent need for a comprehensive solution [4]. This project aims to develop a software application integrating GIS and mapping functionalities to gather, process, and visualize real-time flood data. Hence, this project's objectives are to develop a user-friendly centralized dashboard that provides real-time visualizations of sensor data, enabling users to monitor the flood situation and make informed decisions. Also, it is to develop an automation warning system that is responsive to timely events and can send out on time alerts to residents in flood-prone areas. By addressing these critical shortcomings, the goal is to empower data-driven decision-making, enhance community readiness, and minimize the impact of recurring flood events. Through improved accuracy in predictions and timely alerts to at-risk regions, this initiative strives to significantly reduce the vulnerabilities associated with floods, fortifying the region's resilience in the face of these natural disasters.

2. Related Works

In the background study, we delved into the technical aspects of these systems. We discussed the NodeMCU ESP32 with Wi-Fi as a communication module that enables connectivity in these systems. This module is crucial for transmitting data from sensors to servers or cloud platforms [5]. We also explored Android Studio as a powerful tool for developing mobile applications. Android Studio provides a comprehensive set of tools for building apps on every type of Android device, making it an ideal choice for developing the mobile application component of a smart flood warning system [6]. On the other hand, Firebase Realtime Database is used for data storing service as it is a cloud-hosted NoSQL database that allows data to be stored and synchronized in real-time across all connected clients [7].

In reviewing previous projects, we examined various implementations of smart flood warning systems. Each project provided unique insights into different approaches to system design and highlighted innovative solutions in sensor technologies and mobile communication platforms. For example, the "Flood Early Warning Information System for Multilocation Based Android" by Dedi Satria presents a significant development in the field of flood early warning technology where a prototype model of a flood monitoring system using Android combining with various sensors to detect and monitor flood conditions is designed [8]. The comparison of previous projects allowed us to identify common trends, unique features, and potential areas for improvement in existing systems [9][10][11][12][13]. This comparative analysis is instrumental in understanding the strengths and weaknesses of different systems and informing the development of a more effective solution.

In discussing challenges and opportunities, we acknowledged the difficulties in implementing these systems, such as excessive costs, security concerns, lack of infrastructure, and maintaining network connectivity in adverse weather conditions. However, we also recognized several opportunities presented by new technologies such as real-time monitoring, use of IoT and machine learning technologies for more accurate predictions and timely warnings, development of mobile applications for enhanced accessibility and user-friendliness, use of solar power for sustainable operation, and integration with other disaster management systems for a more comprehensive approach to disaster management.

3. Methodology

This project focuses on creating an Android application interfacing with an ESP32 device to monitor and predict flood conditions. The application will visually represent four types of data—water level, rising rate of water, rainfall intensity, and water flow rate—accessible through distinct buttons on the main interface. Incorporating the Google Maps API, the app will display flood hotspots based on device locations, allowing users to set favourite areas for monitoring potential flood risks. A pivotal feature involves predictive capabilities, utilizing collected data to forecast potential flood events and automatically triggering SMS warnings, enabling proactive preparation and response. The forthcoming sections will delineate the methodologies applied in

system architecture, software tools, and testing procedures, aiming to provide a comprehensive understanding of how these methods effectively contribute to achieving the project's objectives.

3.1 Software Development Life Cycle

The project adopted the Agile Software Development Life Cycle (SDLC) model, renowned for its iterative and adaptive nature. Beginning with requirements gathering, the team defined data types, application features, and user experience expectations [14]. The subsequent design phase encompassed system architecture creation, UI design, and planning for ESP32 device interaction. Implementation involved coding, Google Maps API integration, and SMS warning system setup, followed by rigorous testing—unit, integration, and user acceptance—to ensure functionality. Post-testing and deployment to users occurred, yet maintenance persisted, emphasizing continuous updates for sustained effectiveness and user satisfaction. Agile's flexibility facilitated adaptability, meeting project objectives and user needs. Its iterative approach allowed for ongoing refinement, culminating in a robust and user-centric application [15].

3.2 System Architecture

The system architecture of the Smart Flood Warning System is designed to be robust and efficient, capable of handling real-time data processing and alerting. The architecture can be divided into three main components: Data Collection, Data Processing, and User Interface.

The initial component, referred to as the Data Collection module, is entrusted with the critical task of acquiring pertinent data from the ESP32 device. This device diligently monitors and records four crucial parameters: water level, rate of rise in water level, intensity of rainfall, and water flow rate. Upon successful acquisition, this data is relayed to the second component, the Data Processing module.

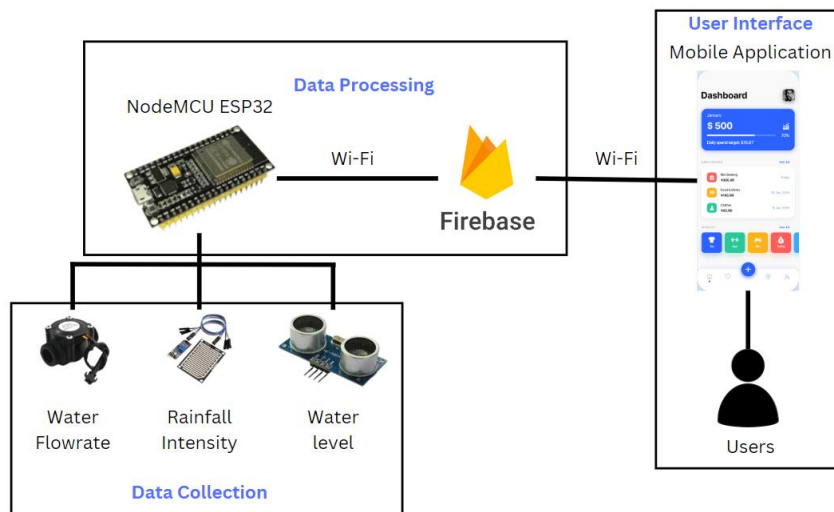


Fig. 1 Project Block Diagram

The Data Processing module, serving as the nucleus of the system, receives the data transmitted from the Data Collection module [16]. It applies sophisticated predictive algorithms to the data, enabling it to forecast potential flood events. If the processed data indicates a potential flood scenario, this module activates the SMS warning system and transmits an alert to the final component, the User Interface.

The User Interface, an Android application, acts as the primary interaction point for the users with the system. It elegantly presents the collected data in a graphical format, pinpoints flood hotspots on a map utilizing the Google Maps API and delivers alerts to users when potential flood events are identified. Furthermore, it offers users the flexibility to set favourite locations for personalized monitoring.

This system architecture fosters efficient data flow and processing, ensuring users are promptly alerted about potential flood events. Its modular design also facilitates easy updates and maintenance, allowing the system to adapt to evolving requirements and conditions.

3.3 Flowchart of the System

The system workflow initiates by configuring and connecting to a Wi-Fi network for seamless communication with external services like Firebase Realtime Database. Sensor data, collected and locally processed within the ESP32, undergo flood prediction algorithms before being sent to Firebase for analysis and storage. The mobile app retrieves this data from Firebase, presenting it graphically for user interaction. Through the app, users can observe flood hotspots on a map, access data, and receive alerts. In the event of an anticipated flood, the system triggers an SMS Warning System, alerting residents and authorities for immediate action. Which the conditions of determining if there is flood event is set as water level higher or equal to 2.3m, rain intensity higher or equal to 30mm/h, and water level rising rate higher or equal to 0.5m/minute. Operating in a continuous loop, the system remains responsive and proactive, perpetually monitoring, processing, and alerting in real-time. The flowchart concludes with the system's ongoing readiness for continuous monitoring and response.

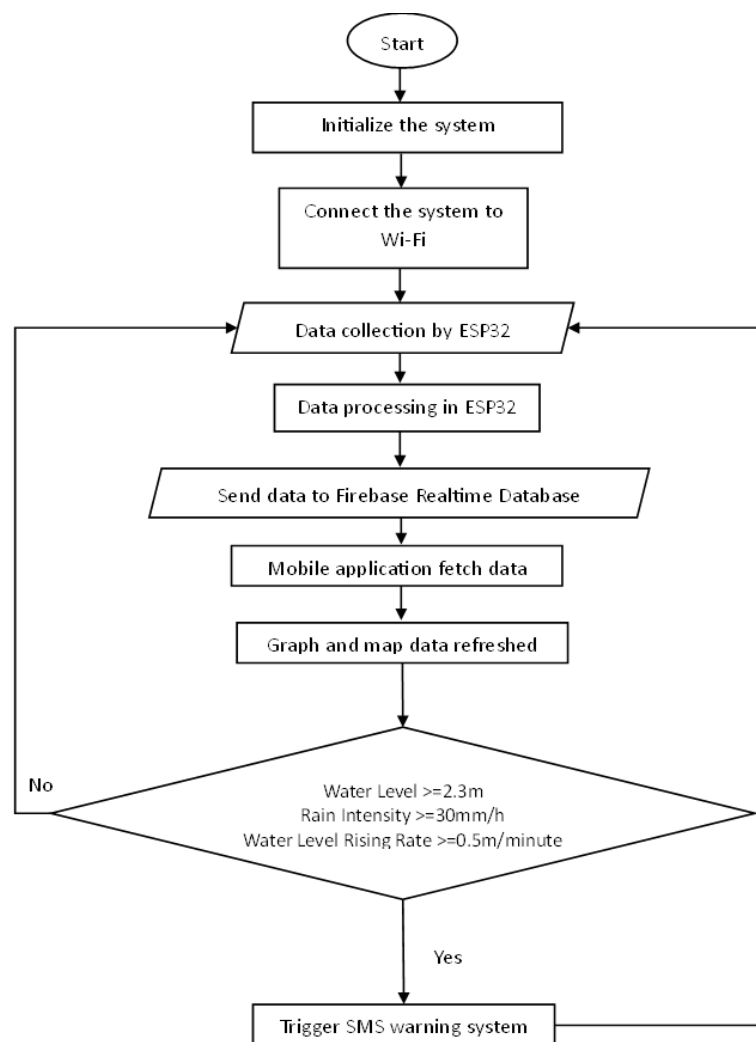


Fig. 2 Flowchart of the System

3.4 User Interface Design

In the initial phase of the design process, a wireframe was created to outline the basic structure and layout of the homepage. The wireframe serves as a low-fidelity blueprint that focuses on the essential elements and their placement [17].

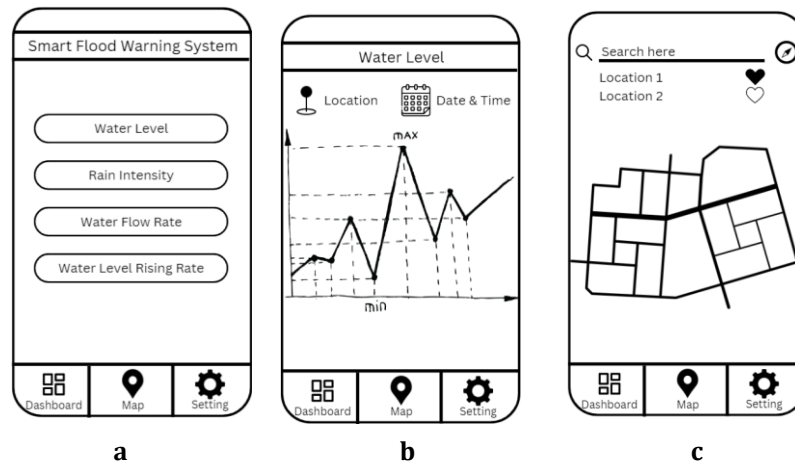


Fig. 3 (a) Wireframe for Homepage (b) Wireframe of graph of water level (c) Wireframe of Map Fragment

The wireframe for the mobile app's homepage prioritizes a clean, user-centric interface. It features a headline at the top, followed by four navigation buttons for easy access to essential features like water level and rainfall intensity graphs. A bottom navigational bar offers quick links to the homepage, a specialized map interface, and customization options. When users select a button, they're directed to dedicated fragments displaying relevant graphs, ensuring efficient access to specific information. A 'Select Location' button allows tailored data viewing, while a 'Choose Time' option offers customizable time intervals. The design aims for simplicity and user control, providing straightforward tools for exploring water-related data.

The Map Fragment provides dynamic visualization of water-related situations, featuring a search bar for location input and a button to locate the user. Smart suggestions streamline location searches, enhancing user experience. Together, these elements offer a user-friendly and interactive experience for exploring water situations in different areas efficiently.

3.5 Database Design

The Smart Flood Warning System employs Firebase Realtime Database, a cloud-hosted NoSQL database, for data logging and retrieval. This database is designed to store and synchronize data in real-time, making it an ideal choice for this application.

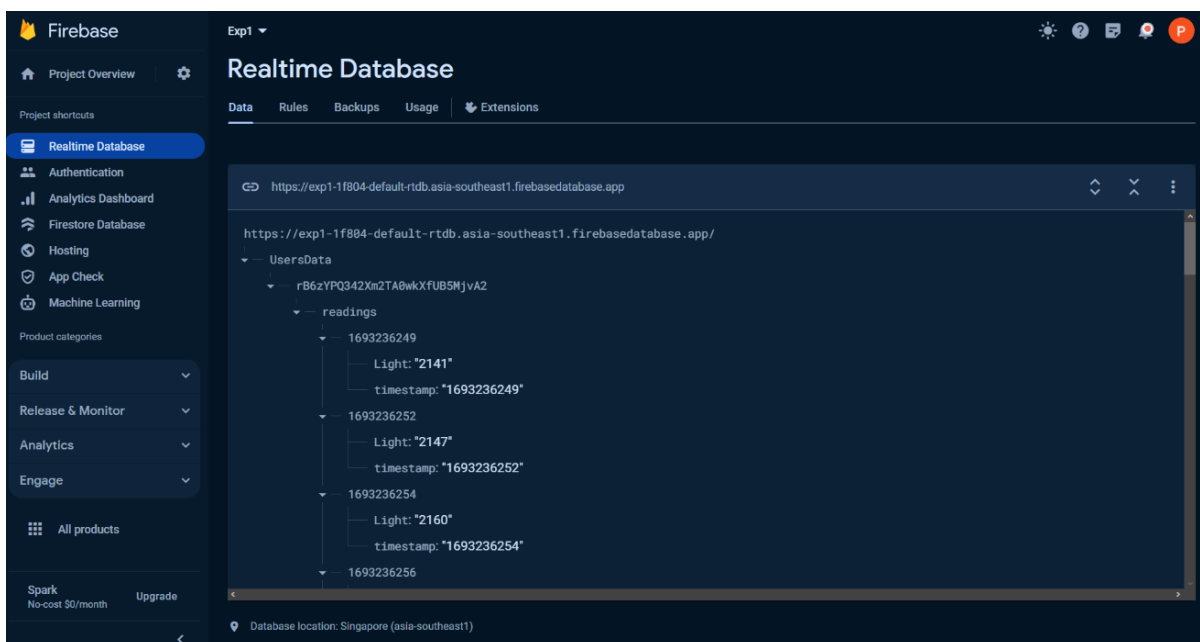


Fig. 4 Firebase Realtime Database Data Logging

The structure of the database is designed to efficiently store the four types of data collected from the ESP32 device: water level, water level rising rate, rainfall intensity, and water flow rate. Each of these data types is stored in a separate node in the database, allowing for efficient data retrieval. The nodes are structured in a way that each data entry is timestamped, ensuring the sequence of data is maintained. This is crucial for accurately representing the progression of flood conditions over time.

The mobile application fetches data from the Firebase Realtime Database to populate the graphs displayed to the user. This ensures that the user is always presented with the most up-to-date information, enabling them to make informed decisions in the event of a potential flood. The use of Firebase Realtime Database not only facilitates real-time data logging and retrieval but also provides a scalable and flexible solution that can adapt to changing data requirements [18].

3.6 Software Used

The project utilized a suite of diverse software tools catering to specific development needs. Microsoft Word, a versatile word processing software, enabled seamless document creation and sharing, offering premium features like advanced grammar and dictation [19]. Arduino IDE facilitated code writing, hardware communication, and program uploads for Arduino hardware devices [20]. Android Studio, the official IDE for Android app development, provided a robust environment with a Gradle-based system, powerful emulator, testing tools, and Google Cloud Platform support [21]. Firebase, a Google product, offered a comprehensive backend service for app development, encompassing databases, authentication, and cross-platform SDKs for various languages and frameworks, enabling easy app management and growth [22]. Each software played a crucial role, from document creation to hardware programming and app development, offering specialized functionalities essential to the project's success.

4. Results and Discussion

4.1 Data Logging via Firebase Realtime Database

The application functionality test focused on validating the data logging capability between the ESP32 microcontroller and the Firebase Realtime Database, employing an LDR sensor for light intensity measurements. The test encompassed the initialization, data collection, and Firebase upload phases. During initialization, the ESP32 seamlessly connected to the Wi-Fi network, and Firebase credentials were configured without errors. The subsequent data collection from the LDR sensor yielded consistent raw analog readings displayed in the serial monitor. The pivotal aspect of the test was the successful upload of light intensity data and timestamps to the Firebase Realtime Database. Each database entry corresponded accurately to its timestamp and contained the associated light intensity value.

```
https://exp1-1f804-default-rtdb.asia-southeast1.firebaseio.com/
├── UsersData
│   └── rB6zYPQ342Xm2TA0wkXfUB5MjvA2
│       └── readings
│           ├── 1693236249
│           │   ├── Light: "2141"
│           │   └── timestamp: "1693236249"
│           ├── 1693236252
│           │   ├── Light: "2147"
│           │   └── timestamp: "1693236252"
│           ├── 1693236254
│           │   ├── Light: "2160"
│           │   └── timestamp: "1693236254"
│           └── 1693236256
```

Fig. 5 Data logging in Firebase Realtime Database

Observations from the test indicated the ESP32's reliable collection of light intensity data and the consistent and accurate recording of entries in the Firebase Realtime Database. Figure 5 shows that the data is stored in the Firebase Realtime Database under the tab, UsersData with the device UID as 'rB6zYPQ342Xm2TA0wkXfUB5MjvA2'. Each of the data value is recorded under its timestamp following the Unix time format. The "Light" in the data recorded means light intensity, is recorded correctly as shown as the data displayed in the Arduino IDE.

Test 1 is data logging for 1 second delay. From Table 1 above, it shows that in a time duration of 20.43s, there are 18 data collected by the ESP32. Among the 18 data, only 10 data is logged into Firebase Realtime Database, while there are 8 data is not recorded. The average time taken for logging data into firebase is about 2.043s. The possible reason for data not sent to firebase is hardware constraint as the microcontroller, ESP32 has limited processing power and memory to handle a high frequency of data transmission.

Table 1 Data Logging Test for 1 second delay

No.	Timestamp	Relative Time (s)	Light Intensity	Data log in Firebase
1	18:57:24.128	0	2928	YES
2	18:57:26.922	2.794	2993	YES
3	18:57:28.162	4.034	2996	NO
4	18:57:29.144	5.016	2977	YES
5	18:57:30.405	6.277	2983	NO
6	18:57:31.412	7.284	2973	YES
7	18:57:32.522	8.394	2998	NO
8	18:57:33.526	9.398	2992	YES
9	18:57:34.704	10.576	2991	NO
10	18:57:35.704	11.576	2986	YES
11	18:57:36.934	12.806	2990	NO
12	18:57:37.948	13.820	2981	YES
13	18:57:39.221	15.093	2986	NO
14	18:57:40.180	16.052	2989	YES
15	18:57:41.481	17.353	2982	NO
16	18:57:42.431	18.303	2983	YES
17	18:57:43.555	19.427	2987	NO
18	18:57:44.558	20.430	3006	YES

Test 2 is data logging test for 1 minute delay. By analysing the data recorded in Table 2, it shows that when the time duration used to log the data into firebase is set to 1 minute, 10 out of 10 data is successfully recorded in the database. The test indicated that the data logging function is successfully implemented by the Smart Flood Warning System.

Table 2 Data Logging Test for 1 minute delay

No.	Timestamp	Relative Time (minute)	Light Intensity	Data log in Firebase
1	19:37:34.013	0	1062	YES
2	19:38:34.674	1	889	YES
3	19:39:34.885	2	962	YES
4	19:40:35.229	3	983	YES
5	19:41:36.237	4	869	YES
6	19:42:36.483	5	865	YES
7	19:43:36.663	6	944	YES
8	19:44:37.192	7	982	YES
9	19:45:37.511	8	957	YES
10	19:46:37.708	9	913	YES

4.2 Data Visualization in Mobile Application

The utilization of the AnyChart API in conjunction with Firebase Realtime Database proved instrumental in visualizing dynamic datasets within the project. Through the successful implementation of the AnyChart API, data from the Firebase Realtime Database was seamlessly translated into visually informative graphs. These visual representations provided an effective means to comprehend and analyse the streamed data.

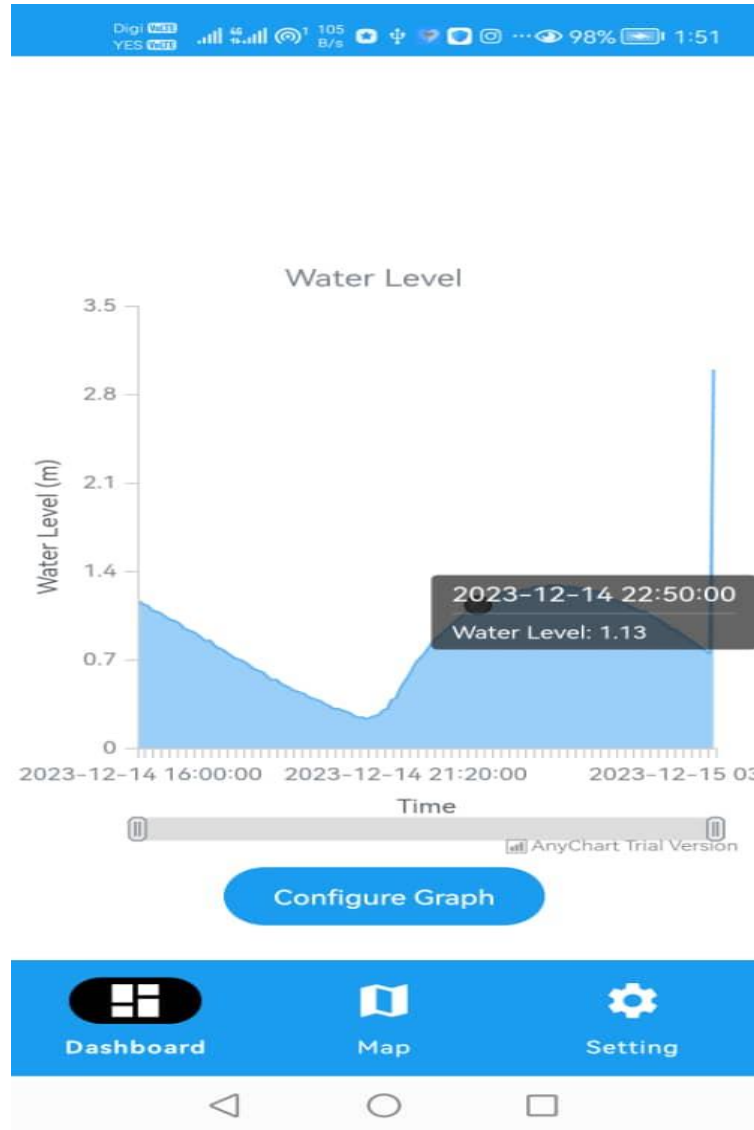


Fig. 6 Water Level against Time Graph

Figure 6 shows one of the graphs from the Smart Flood Warning System which is the Water Level against Time Graph. The figure above indicated that the data visualization is perfectly implemented into the mobile application. By using the AnyChart library, we can look for graphs detail by simply press on the graph and there will be a tooltip displaying the collected time of the data and the value of it. The user also can zoom in and out the graph by adjusting the scroller below the chart.

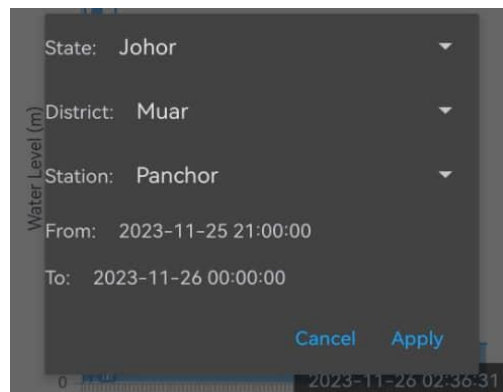


Fig. 7 Configuration Dialog



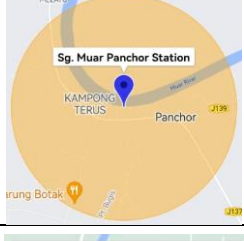

A key feature integrated into the visualization module is the configuration settings button, empowering users to define specific parameters—such as state, district, station, and time range—to tailor the displayed graph. Upon applying the selected settings, the graph dynamically refreshes and updates to present the dataset matching the newly configured parameters. This dynamic refresh functionality ensures that the graph accurately reflects the user-defined criteria, thereby enhancing the adaptability and relevance of the displayed data.

The incorporation of configuration settings augments user interaction, allowing for a personalized data exploration experience. Users can seamlessly adjust the parameters to explore distinct data subsets, fostering a deeper understanding of trends, patterns, and correlations within the dataset. The user-driven customization capabilities empower individuals to efficiently extract relevant insights specific to their analytical needs, thereby amplifying the utility and value of the visualization tool.

4.3 Hotspot Area in Map Fragment

The Map Fragment incorporates a crucial functionality to visualize the severity of potential floods through a dynamic hotspot area feature. This innovative addition enables users to discern the severity levels based on color-coded indications within specific geographic areas. The hotspot areas on the map dynamically change colour based on the severity predictions calculated for potential floods. The algorithm underlying this functionality evaluates multiple parameters, including historical data, current environmental conditions, and predictive models, to forecast the severity of potential flooding events. These predictions are then translated into colour variations within the hotspot areas, providing users with an immediate visual indication of the anticipated flood severity levels within specific regions. The water level condition in Table 3 is set according to the reference from InfoBanjir by Water Resources and Hydrology Division, Department of Irrigation and Drainage, Malaysia [23]. Which for Muar River at Panchor Station, the water level below 2.0m is classified as normal, below 2.3m is alert, water level below 2.5m as warning and lastly the danger water level is higher than 2.5m.

Table 3 Hotspot Area

No.	Condition	Colour	Picture
1	Water Level < 2.0m	Green	
2	Water Level < 2.3m	Yellow	
3	Water Level < 2.5m	Orange	
4	Water Level >= 2.5m	Red	

The color-coded representation of flood severity levels enhances the user interface by providing an easily understandable and actionable visual indicator. Users can quickly interpret the severity levels indicated by different colours within the hotspot areas, facilitating informed decision-making in disaster management and emergency response strategies. The intuitive nature of this visualization aids in swiftly assessing the urgency and scale of potential flood risks within specific regions.

4.4 SMS Warning Alert Functionality

The Smart Flood Warning System is a critical application designed to keep users informed about potential flood events. Central to its functionality is the SMS Warning Alert system, a vital component ensuring timely notifications. A successful test scenario was conducted to validate this feature's effectiveness in alerting users under various conditions. The test proceeded with simulating flood conditions by manipulating the monitored Firebase data. Changes in water levels, rain intensity, or rising water levels were adjusted to meet predefined flood conditions. The system's performance was gauged by checking if an SMS alert was successfully dispatched to the configured phone number. The content of the SMS was thoroughly inspected to ensure it contained relevant flood event details, such as water level information and a clear warning message.

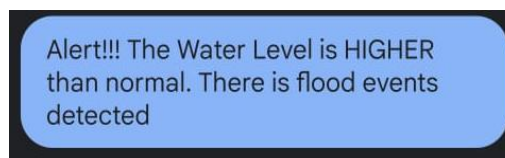


Fig. 8 SMS 1

The alert shown in Figure 8 is triggered when the water level surpasses or equals 2.3, signifying a potential flood event. The message content reads, "Alert!!! The Water Level is HIGHER than normal. There are flood events detected." This SMS aims to promptly notify users about the escalated water levels, prompting them to be cautious and prepared for potential flooding.

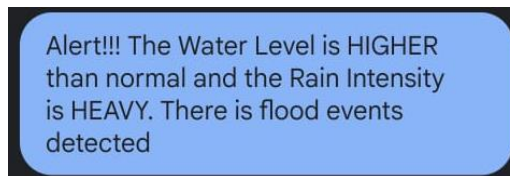


Fig. 9 SMS 2

When the rain intensity reaches or exceeds 30 and the water level is at or above 2.0, an alert is generated as shown in Figure 9. The message content states, "Alert!!! The Water Level is HIGHER than normal, and the Rain Intensity is HEAVY. There are flood events detected." This notification serves as a critical warning, indicating heavy rainfall contributing to the rising water levels, alerting users to an imminent flood risk.

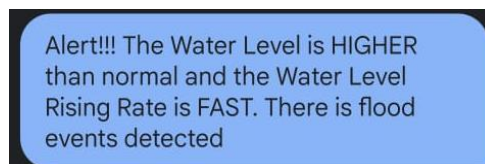


Fig. 10 SMS 3

Additionally, negative scenarios were tested. The phone number was deliberately removed or invalidated in the settings, and the alert was triggered again. As expected, no SMS alert was sent in this scenario, validating the system's response to missing or invalid phone numbers.

Table 4 serves as a comprehensive evaluation matrix for the SMS functionality integrated within the flood warning system. Each entry within the table delineates a unique combination of conditions involving water level thresholds, rain intensity, water level rising rates, and the presence of a configured phone number designated for SMS alerts. The column "SMS Sent" provides an indication of whether the system dispatched an SMS alert based on the specified conditions.

Table 4 SMS Functionality Test

No.	Water Level Condition (m)	Rain Intensity Condition (mm/h)	Water Level Rising Rate (m/minute)	Phone Number Set	SMS Sent
1	≥ 2.3	< 30	< 0.5	YES	SMS 1
2	≥ 2.3	≥ 30	< 0.5	YES	SMS 1
3	≥ 2.3	< 30	≥ 0.5	YES	SMS 1
4	≥ 2.3	≥ 30	≥ 0.5	YES	SMS 1
5	≥ 2.0	< 30	< 0.5	YES	NO
6	≥ 2.0	≥ 30	< 0.5	YES	SMS 2
7	≥ 2.0	< 30	≥ 0.5	YES	SMS 3
8	≥ 2.0	≥ 30	≥ 0.5	YES	SMS 2
9	< 2.0	< 30	< 0.5	YES	NO
10	< 2.0	≥ 30	< 0.5	YES	NO
11	< 2.0	< 30	≥ 0.5	YES	NO
12	< 2.0	≥ 30	≥ 0.5	YES	NO
13	≥ 2.3	< 30	< 0.5	NO	NO
14	≥ 2.0	≥ 30	< 0.5	NO	NO
15	≥ 2.0	< 30	≥ 0.5	NO	NO

Overall, the test confirmed the SMS Warning Alert Functionality's effectiveness within the Smart Flood Warning System. It demonstrated a capability to deliver precise and timely notifications during potential flood events while showcasing resilience in handling both expected and unexpected scenarios.

5. Conclusion

The successful execution of the Mobile Application for the Smart Flood Warning System fulfills its primary objectives: creating an intuitive dashboard for real-time sensor data visualization and implementing an automated alert system for timely warnings in flood-prone areas. Achieving this involved leveraging Firebase Realtime Database for seamless data logging and the AnyChart API for user-friendly data visualization, ensuring reliable data collection and presentation. The system's Hotspot Area feature dynamically illustrates flood severity on a map, aiding proactive planning. Additionally, the SMS Warning Alert Functionality demonstrated remarkable reliability in dispatching precise alerts during potential flood events, showcasing adaptability and resilience in varying environmental conditions. Overall, these accomplishments collectively enhance user understanding, proactive decision-making, and efficient response to flood-related events.

5.1 Recommendations

Enhancements to the system involve integrating GSM technology alongside Wi-Fi for broader coverage, especially in remote areas, ensuring failsafe communication vital for timely alerts. Additionally, incorporating a Telegram bot expands outreach, engaging users directly through the Telegram application. Integrating machine learning algorithms refines flood prediction accuracy by analyzing historical data, while a robust feedback mechanism encourages community engagement. Lastly, implementing multilingual support enhances user comprehension of alerts, collectively aiming to fortify reliability, predictive capabilities, and user engagement for more effective flood management.

Acknowledgement

I extend my deepest gratitude to the University Tun Hussein Onn Malaysia for providing me with the platform and resources essential for the completion of this project on "Development of Mobile Application for Smart Flood Warning System." My heartfelt appreciation goes to my supervisor, Encik Mohd Fadly Bin Abd Razak, whose guidance, insights, and unwavering support were invaluable throughout this endeavour. His expertise and encouragement have been instrumental in shaping this work. I am also immensely thankful to my parents for their boundless love, encouragement, and unwavering belief in my abilities. Their endless support and sacrifices have been the cornerstone of my academic journey. Lastly, I extend my thanks to all those who, directly or indirectly, contributed to this project and helped me in any form.

References

- [1] Wang, X.; Xie, H. A review on applications of remote sensing and geographic information systems (GIS) in water resources and flood risk management. *Water* 2018, 10, 608.
- [2] Haq, M., et al. (2012). Techniques of remote sensing and GIS for flood monitoring and damage assessment: A case study of Sindh province, Pakistan. *The Egyptian Journal of Remote Sensing and Space Science*, 15(2), 135–141.
- [3] Samansiri, S.; Fernando, T.; Ingirige, B. Advanced Technologies for Offering Situational Intelligence in Flood Warning and Response Systems: A Literature Review. *Water* 2022, 14, 2091. <https://doi.org/10.3390/w14132091>
- [4] Sadiq, R., Imran, M., Ofli, F. (2023). Remote Sensing for Flood Mapping and Monitoring. In: Singh, A. (eds) *International Handbook of Disaster Research*. Springer, Singapore. https://doi.org/10.1007/978-981-16-8800-3_178-1
- [5] CircuitSchools (2022). What is ESP32, how it works and what you can do with ESP32? Retrieved on October 15, 2023, from <https://www.circuitschools.com/what-is-esp32-how-it-works-and-what-you-can-do-with-esp32/>
- [6] Android Developers (2023). Meet Android Studio. Retrieved on October 15, 2023, from <https://developer.android.com/studio/intro>
- [7] Patel, U. (2016). Why Firebase is the Best Mobile Backend as a Service? Tristate Technology. Retrieved on November 1, 2023, from <https://www.tristatetechnology.com/blog/firebase-backend-mobile-app>
- [8] Satria, D., Yana, S., Munadi, R., & Syahreza, S. (2020). FLOOD EARLY WARNING INFORMATION SYSTEM FOR MULTILLOCATION BASED ANDROID. *International Journal of Engineering Technologies and Management Research*, 5(8), 47–53. <https://doi.org/10.29121/ijetmr.v5.i8.2018.279>
- [9] Shylesh, S. (2017). A study of software development life cycle process models. *Social Science Research Network*. <https://doi.org/10.2139/ssrn.2988291>
- [10] Sufa, H. F. A., Yusof, M. I. & Sani, M. A. A. (2019). FLOOD MONITORING AND WARNING SYSTEM WITH IOT. *Malaysian Journal of Industrial Technology*, Volume 3, No. 2, 2019.
- [11] Zakaria, M. I., Jabbar, W. A., & Sulaiman, N. (2023). Development of a smart sensing unit for LoRaWAN-based IoT flood monitoring and warning system in catchment areas. *Internet of Things and Cyber-Physical Systems*, 3, 249–261. <https://doi.org/10.1016/j.iotcps.2023.04.005>.
- [12] Ali, S. A., Ashfaq, F., Nisar, E., Azmat, U., & Zeb, J. (2020). A Prototype for Flood Warning and Management System using Mobile Networks. 2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST). <https://doi.org/10.1109/ibcast47879.2020.9044531>
- [13] Bentoso, L. D., Juan, E. O., Brosas, D. G., Paragas, J. R., Nuevas, L. K., & Velarde, M. W. C. (2021). Web-Based Solution for Flood Warning Decision Support in the Province of Leyte, Philippines. *International Conference on Research and Academic Community Services (ICRACOS)*. <https://doi.org/10.1109/icracos53680.2021.9701990>
- [14] Kramer, M. (2018). Best Practices in Systems Development Lifecycle: An analyses based on the Waterfall model. *Review of Business and Finance Studies*, 9(1), 77–84. <https://EconPapers.repec.org/RePEc:ibf:rbfstu:v:9:y:2018:i:1:p:77-84>
- [15] Affandi, R., Omar, Z., & Lee, M. F. (2023). DEVELOPMENT OF THE HEADWATER EARLY SIGNS BY USING INTERNET OF THINGS (IoT). *International Journal of Engineering Advanced Research*, 5(2), 61-71.
- [16] de Paula, D.F.O., Menezes, B.H.X.M., Araújo, C.C. (2014). Building a Quality Mobile Application: A User-Centered Study Focusing on Design Thinking, User Experience and Usability. In: Marcus, A. (eds) *Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments*. DUXU 2014. Lecture Notes in Computer Science, vol 8518. Springer, Cham. https://doi.org/10.1007/978-3-319-07626-3_29
- [17] Soegaard, M. (2023, December 25). How to create wireframes: an expert's guide. The Interaction Design Foundation. <https://www.interaction-design.org/literature/article/create-wireframes>
- [18] Maulana, I. F. (2020). Penerapan Firebase Realtime Database pada Aplikasi E-Tilang Smartphone berbasis Mobile Android. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 4(5), 854–863. <https://doi.org/10.29207/resti.v4i5.2232>
- [19] Writtenhouse, S. (2022, June 21). *7 Awesome Microsoft Word features you should be using*. How-To Geek. <https://www.howtogeek.com/807229/word-features-you-should-be-using/>

- [20] Adnanaqeel. (2021, July 31). *Introduction to Arduino IDE*. The Engineering Projects. <https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html>
- [21] *Meet Android Studio*. (n.d.). Android Developers. <https://developer.android.com/studio/intro>
- [22] Omisola, I. (2021, December 5). *What is Google Firebase and why should you use it?* MUO. <https://www.makeuseof.com/what-is-google-firebase-why-use-it/>
- [23] National Flood Forecasting and Warning Centre (PRABN), Water Resources and Hydrology Division, Department of Irrigation and Drainage, Malaysia (2019). PUBLIC INFOBANJIR. Retrieved on November 1, 2023, from <https://publicinfobanjir.water.gov.my/mengenai-kami/publicinfobanjir/?lang=en>