Chapter I

The Problem and its Background

Introduction

The Safety on campuses continues to be a major issue in the world of higher education. Studies have recently shown that there are thousands of accidents taking place in schools each year and that these accidents are usually caused by the neglect of hazards like improper installation of electrical wires, walking paths that are too slippery and fire places that are too close to flammable materials. Accidents like these pose a danger to students and staff, but they also interrupt the academic process, and thereby, cause a depletion of the institution's resources. Pampanga State University (formerly Don Honorio Ventura State University) is a place where every day, thousands of students, faculty, and non-teaching staff share the same space, and therefore the ever-present danger of both natural and artificial disasters cannot be completely avoided. Still, there are safety rules in place, but in fact many accidents and risks are not officially recognized or resolved. One of the causes for that is the old-fashioned reporting system, which mainly involves paper forms, manual submissions, or communication that is too slow. Such a system often ends up in delayed response times, reduced accountability, and accidents that could have been avoided. However, digital fixes such as Progressive Web Applications (PWAs) have the potential to revolutionize the whole idea of risk reporting and hazard management thanks to the rapid technological advancements. PWAs are associated with their fast loading time, the ability to work on different platforms, and synchronization of real-time data, all of which attributes make them suitable for emergency communication and monitoring

of safety. Accordingly, the present research presents RISKWISE—a Progressive Web Application that is intended for Pampanga State University. The application permits the users to report risks through geotagging, incident narratives, and photographs. Submission of reports is possible even when there is no internet connection and such reports are automatically synced when the user is back online. An interactive dashboard is made accessible for the administrators and safety officers to categorize hazards, scale their seriousness, and take preventive actions.

Objective of the Study

To design and implement a Progressive Web Application that improves real-time risk assessment and hazard reporting within Pampanga State University.

objectives:

1. To design a user-friendly reporting interface for submitting incidents with descriptions, geotagging, and photographic evidence.
2. To implement a barcode scanning feature for easy product tracking.
3. To develop an administrator dashboard for monitoring, classifying, and resolving reported hazards.
4. To integrate real-time push notifications for safety updates and critical alerts.
5. To evaluate the system's performance and user satisfaction using the ISO/IEC 25010 software quality model, focusing on functionality, usability, reliability, and performance efficiency.

Significance of the Study

This study will benefit the following stakeholders:

University Administration & Safety Officers: Provides a streamlined monitoring tool for faster decision-making and accurate documentation.

Students, Faculty, and Staff: Empowers users to actively participate in campus safety by reporting risks conveniently.

Developers and IT Professionals: Serves as a practical case study for applying PWAs to institutional problem-solving.

Future Researchers: Offers a foundation for exploring AI-driven hazard detection, automated emergency response, or SMS-based safety alerts.

Academic Community: Demonstrates how digital platforms can be adapted to strengthen safety protocols across educational institutions.

Scope and Limitation

Scope

This study focuses on the design, development, and evaluation of RISKWISE within Pampanga State University. The system includes: Web-based submission of risk reports with geolocation and image attachments. Administrator dashboard for classifying and responding to incidents. Real-time functionalities using modern web technologies. Evaluation using ISO/IEC 25010 standards for quality assurance.

Limitation

The system will not include native mobile applications (Android/iOS). Real-time dashboard updates still require internet connectivity. Automated emergency response (e.g., notifying first responders) is beyond the project scope. Hazard evaluation relies on predefined models, not AI/ML. The system's effectiveness depends on active user participation and accuracy of reports. IoT-based hazard detection hardware is excluded from this version.

Definition of Terms

Risk Assessment: Identifying, analyzing, and evaluating hazards to determine their impact and necessary response.

Incident Reporting: Documenting and communicating details of hazards or safety breaches.

Progressive Web App (PWA): A web application that behaves like a native app, and push notifications.

Geolocation: The process of determining a user's or event's physical location using GPS or similar technology.

Service Workers: Background scripts that enable PWAs to handle caching, and notifications.

IndexedDB: A browser-based database for storing large amounts of structured data.

Push Notification: Push notifications are implemented in the RiskWise system to deliver real-time alerts and updates to users regarding reported hazards, status changes, and important safety announcements. These notifications allow users to receive timely information even when the application is inactive, ensuring continuous awareness of campus safety conditions. The use of push notifications improves communication efficiency between administrators and end-users by minimizing response delays and increasing user engagement with the system.

Chapter II

Review of Related Literature and Studies

Related Literature

Real-Time Risk Assessment and Institutional Safety

In educational institutions, safety is an operational priority. According to Hernandez and Zhao (2022), real-time risk detection systems significantly reduce injuries and property damage in large-scale environments like schools and universities. Their study demonstrated that digital systems allowed for the instant identification of hazardous conditions, followed by faster intervention from administrative units. This is relevant to RISKWISE because the system is specifically built to improve real-time awareness and action at DHVSU through instant digital risk reporting.

Kim, Lee, and Park (2023) investigated the role of real-time feedback systems in emergency situations and concluded that these systems enhanced response outcomes by shortening the decision-making time of safety officers. Although focused on hospitals, their findings validate RISKWISE's core concept: an interactive system that allows safety administrators to respond quickly to incident reports submitted by students, staff, or faculty.

These studies support the idea that risk-related incidents if detected early can be resolved before escalating into full emergencies. RISKWISE contributes to this framework by enabling proactive rather than reactive safety management..

Digital Incident Reporting Tools and Systems

Ignaco (2019, cited in Blessing, 2024) developed a mobile app tailored for school-based incident reporting. Users were able to report broken equipment, unsafe facilities, or bullying incidents through mobile forms. The app's usage led to a documented increase in reported cases, which informed policy improvements. This system mirrors RISKWISE in its user-focused design and its emphasis on visual and geolocation data. The success of this project demonstrates the effectiveness of digital platforms in collecting safety data from a distributed user base. Tan et al. (2020) evaluated a mobile app created for storm preparedness and risk communication. Their study revealed that visual reports (e.g., pictures of blocked drainage or damaged facilities) submitted by citizens resulted in faster municipal response. This strongly aligns with RISKWISE's media attachment feature, which allows users to include photos of hazards on campus. Such visual evidence enhances report credibility and assists administrators in understanding the severity of the situation. Fazzini et al. (2022), while focused on mobile bug reporting in software development, highlighted the importance of capturing accurate, structured, and time-bound feedback through digital systems. The relevance of this finding to RISKWISE lies in the app's emphasis on structured forms and timestamped reports, allowing better tracking of incident history and administrative accountability. These sources reinforce the value of digital platforms in improving the quality, frequency, and reliability of incident reports—core principles on which RISKWISE is built.

Use of Progressive Web Applications (PWAs)

Malhotra (2023) described PWAs as a breakthrough in web development, enabling cross-platform access and responsiveness without the high cost of native mobile app development. These apps load quickly, support offline interaction, and work on any device with a browser. This study directly supports RISKWISE's design choice, as the platform is built as a PWA to ensure accessibility for all DHVSU community members regardless of device type. Fauzan et al. (2021) provided a literature review on PWA deployment in public systems and highlighted the importance of features like background syncing, service workers, and push notifications. These technologies are central to RISKWISE's architecture: the app uses service workers and IndexedDB to support offline reporting, while push notifications keep users updated on risk advisories even when the app is inactive. Furthermore, Blessing (2024) found that users tend to adopt PWAs more readily when compared to traditional apps due to reduced storage requirements and faster access. This makes PWAs ideal for school environments where not all users may have access to high-end smartphones or large data plans. This insight justifies the system's focus on lightweight, browser-based operation. These findings validate the technological choice behind RISKWISE and show that PWAs are an appropriate platform for a scalable, maintainable university-wide system.

Geolocation and Mapping in Risk Management

Lin, Yang, and Stack (2022) conducted a study on geolocation use in disaster response systems. They found that digital tools with geolocation capabilities improved the accuracy and response time of responders by pinpointing exact hazard locations. RISKWISE incorporates this by attaching geolocation metadata to every user report, enabling DHVSU safety personnel to identify problem areas faster.

Navarro de Corcuera et al. (2022) analyzed various mobile apps used in disaster mitigation. Their results suggested that location-based features, when combined with timestamped logs and photos, enhanced the traceability of incidents and streamlined emergency procedures. This supports the concept behind RISKWISE's geotagged photo documentation, which serves both as real-time intelligence and a digital log of recurring safety issues across the university campus.

These studies affirm the importance of integrating location data into risk management systems and justify the inclusion of such features in the RISKWISE system.

Related Studies

iReport App (2021)

iReport is a university-based mobile app launched in 2021 that allows students to report broken facilities or misconduct. The system includes photo evidence and location tagging. Although iReport is Android-only, it laid the foundation for mobile incident reporting in academic settings. RISKWISE improves upon iReport by being a PWA, accessible via any browser and functional even without internet connectivity

.

SafeCampus Web Portal (2022)

Developed for disaster risk reduction in a state college, SafeCampus features web-based risk documentation, admin dashboards, and severity scoring. However, it lacks mobile compatibility and offline reporting. RISKWISE addresses these gaps by integrating mobile-first responsive design and offline-first architecture, making it more accessible and resilient

.

LGU RiskMap (2023)

RiskMap is a municipal application used by barangay officials to log flooding, fire, and structural damage reports. It has mapping and analytics tools similar to those in RISKWISE. However, RiskMap is heavily reliant on real-time internet connection. RISKWISE offers a strategic advantage with its offline reporting and syncing capability, crucial in low-connectivity campus zones.

IncidentEye (2021)

Another notable system is IncidentEye, a commercial mobile application developed by Reactec Ltd. In 2021. Designed for enterprise and lone-worker safety, IncidentEye includes advanced features such as GPS tracking, panic buttons, live location sharing, and real-time alert escalation. Its interface supports both mobile and web platforms, and it is primarily used in professional environments where real-time monitoring of personnel is critical. While IncidentEye is robust and effective in its intended domain, it is tailored for corporate and industrial use cases, often requiring dedicated support teams and subscription-based access. In contrast, RISKWISE adapts some of the practical elements of IncidentEye, such as real-time alerting and geolocation, but simplifies the approach for a university environment. RISKWISE is intentionally designed to be user-friendly, cost-effective, and accessible to non-technical users such as students and faculty, without compromising on essential safety reporting capabilities.

eBayanihan App (2020)

The eBayanihan App, developed by the UP Resilience Institute in 2020, was another timely system deployed during the COVID-19 pandemic. It allowed users to report symptoms, crowding, and community-level hazards using a web-based interface that required no installation. The system included geolocation tracking, real-time submission, and data aggregation features that enabled health authorities to monitor and respond to pandemic risks efficiently. eBayanihan's design focused on

community-sourced data and rapid deployment across municipalities. This aligns with RISKWISE's objective of gathering hazard information from a broad range of users, students, staff, and safety officers—to inform institutional safety strategies. While eBayanihan focused primarily on health risks, RISKWISE expands this idea by supporting a wider range of risk types, including infrastructure damage, electrical hazards, and unsafe campus conditions. Additionally, RISKWISE adds offline functionality and customizable scoring, features not present in eBayanihan but critical for resilience in varying connectivity environments.

System Technical Background

The focus of this study is to create an automated watering device with multiple sensor such as humidity sensor, temperature sensor and soil moisture sensor using Internet of Things (IoT) technology and a mobile application for seedlings producers and farmers. The goal is to develop a smart solution that allows producers to track and control the vital factors of humidity, temperature, and soil moisture necessary for the well-being and development of the seedlings.

Conceptual Framework

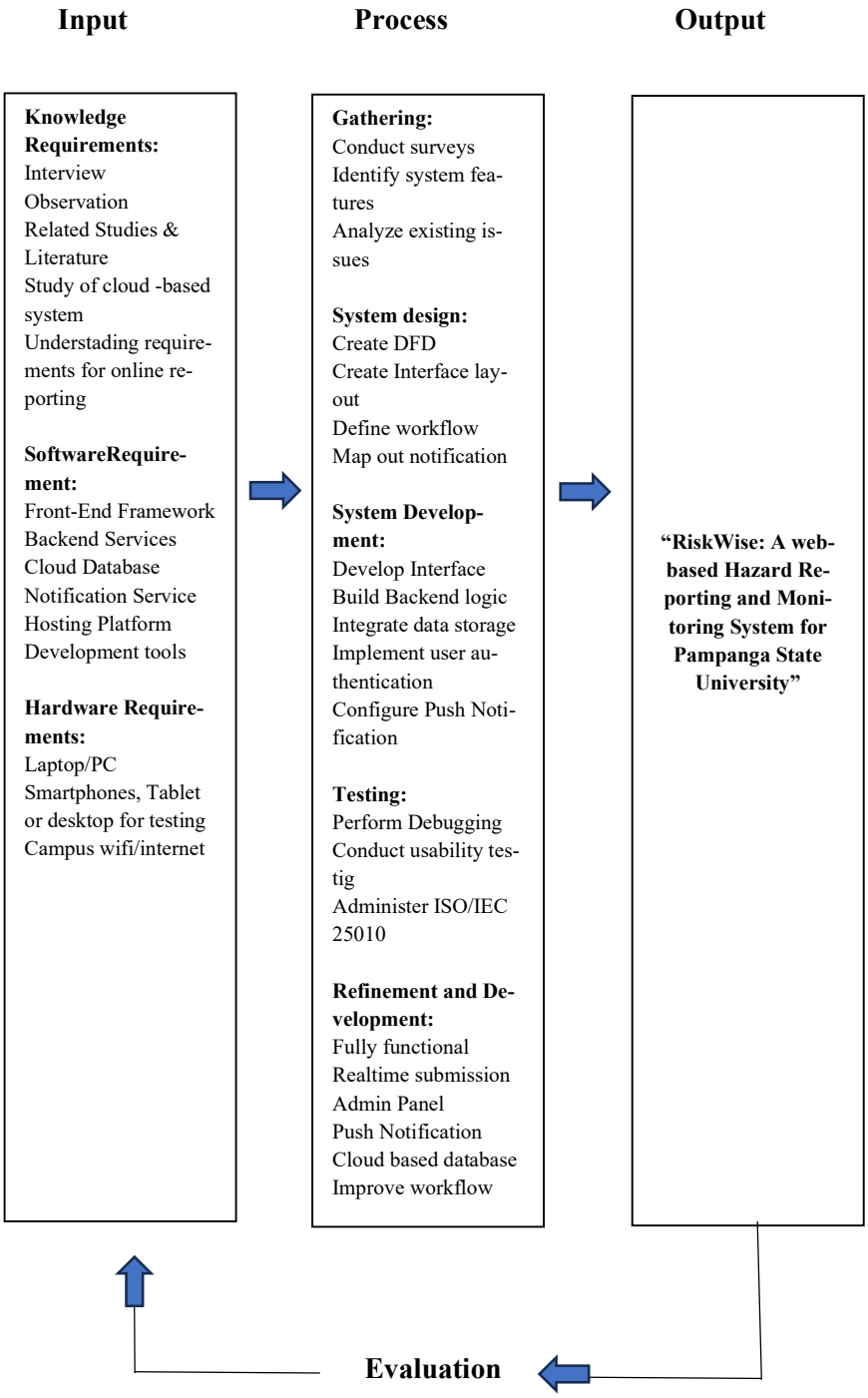


Figure 1. *Conceptual Framework*

The conceptual framework illustrates the way in which the RiskWise Hazard Reporting System was developed using the essential inputs and transforming them into a functional output through a structured process. The inputs, that is, gathered requirements, technical tools, and user needs, form the basis of system planning. These inputs go through systematic procedures that consist of requirement analysis, system design, development, testing, and evaluation. The project subsequently yields the final output, which is a fully functional web-based hazard reporting system that enhances the reporting, monitoring, and managing of campus safety concerns. Thus, the IPO model clearly shows how each phase is crucial for the development of an efficient and quick-response safety reporting solution.

CHAPTER III

METHODOLOGY

Chapter III presents the research design, development process, data gathering procedures, and evaluation methods used in the study. This chapter explains how the RiskWise Hazard Reporting System was designed, developed, tested, and evaluated using appropriate software engineering practices and research techniques. It discusses the research design employed, the sampling and data gathering procedures, the system evaluation framework based on the ISO/IEC 25010 software quality model, and the software development methodology used in building the system. The purpose of this chapter is to provide a clear and systematic description of the processes undertaken to ensure the reliability, effectiveness, and quality of the developed system.

Research Design

The researchers used a mixed-method descriptive developmental research design. This design was chosen because the study required both the creation of a technological solution and the evaluation of its effectiveness within an academic institution. The descriptive developmental aspect came into play during the conceptualization and development of RISKWISE. At the same time, quantitative and qualitative data complemented the evaluation of the system.

The qualitative part mainly involved interviews and informal conversations with students, faculty, administrative staff, and campus safety personnel. These discussions supplemented the team's understanding of how hazard reporting currently works at the university, the issues typically encountered, and the expectations for a modernized system. These qualitative insights were not statistically treated but were used mainly as foundations for the system requirements.

The quantitative aspect focused on the evaluation of the system. Respondents rated the quality of RISKWISE using ISO/IEC 25010 dimensions through a structured Likert-scale questionnaire. The researchers then used these ratings to compute weighted means and overall quality indicators.

This quantitative evaluation was essential in determining whether the system met accepted levels of software quality, particularly in terms of Functionality, Usability, Performance Efficiency, Compatibility, Reliability, Security, Maintainability, and Portability.

This mixed approach was instrumental in ensuring that the developed system was grounded in real institutional needs while still meeting globally accepted standards of software quality.

Respondents of the Study

The respondents of the study consisted of two groups: end-users and IT experts. The end-users included students, faculty members, and non-teaching staff of Pampanga State University who were potential users of the RiskWise system. The

IT experts were professionals with experience in software development and system evaluation, tasked with assessing the technical quality of the system.

Sampling Technique

Simple random sampling was used to select end-user respondents to ensure equal opportunity of participation among members of the university community. Purposive sampling was applied in selecting IT experts, as their technical background and expertise were essential in evaluating the system's maintainability, security, and overall technical performance.

Sampling Technique

- **Simple Random Sampling** was used to select end-users among residents and barangay staff.
- **Purposive Sampling** was used to select IT experts for the technical evaluation of the system.

Respondents of the Study

1. **End-users**: Students, Faculty and Non-Faculty of Pampanga State University.
2. **IT Experts**: Evaluated the technical performance and quality.

Data Gathering Procedure

- **Qualitative Data Collection -** Initial data were gathered through interviews and open-ended questionnaires to understand the issues in managing barangay records and services.
- **Quantitative Data Collection -** After system development, evaluation was conducted through a 4 Likert-scale questionnaire based on ISO/IEC 25010 Software Quality Assurance. The instrument was distributed to both End-users and IT experts.

System Evaluation Framework

The system was evaluated based on ISO/IEC 25010 with the following Criteria:

- **Functional Suitability**

  The evaluation results indicate that the RiskWise system demonstrates strong functional suitability. Respondents agreed that the system provides all the necessary features required for effective hazard reporting within the university. Users found that submitted information is accurately processed and properly displayed, ensuring that reported hazards are clearly communicated to administrators. Additionally, the system was perceived as capable of meeting the campus safety reporting needs by supporting timely submission, tracking, and management of incidents. These findings suggest that RiskWise successfully fulfills its intended purpose and delivers the core functionalities required for a hazard reporting and monitoring system.

- **Performance Efficiency**

  Results show that the RiskWise system performs efficiently during regular use. Users reported that system pages load quickly and that the system responds promptly to user actions such as report submission, navigation, and data retrieval. This indicates that the system is capable of handling user interactions without noticeable delays. The efficient performance contributes to a smoother user experience and supports real-time reporting requirements, which are critical in addressing campus safety concerns.

- **Compatibility**

  The system demonstrated good compatibility across different platforms and environments. Respondents indicated that RiskWise functions properly on commonly used web browsers and does not conflict with various device configurations. This suggests that users can access the system using different devices such as laptops, desktops, and mobile phones without encountering major technical issues. The compatibility of the system ensures broader accessibility and usability across the university community.

- **Usability**

  The evaluation results reveal that RiskWise is user-friendly and easy to learn, particularly for first-time users. Respondents agreed that the system interface is clear, understandable, and well-organized. Icons, buttons, and labels were found to be meaningful and intuitive, allowing users to navigate the system

with minimal difficulty. These findings indicate that the system design effectively supports user interaction and reduces the learning curve, which is essential for encouraging active participation in hazard reporting.

- **Reliability**

  The system exhibited a high level of reliability based on user feedback. Respondents agreed that RiskWise operates without frequent crashes or errors and that submitted reports are saved accurately and securely. The consistent performance of the system assures users that their reports are properly recorded and retained. This reliability is crucial for maintaining trust in the system and ensuring continuous operation during regular and high-demand usage.

- **Security**

  Security emerged as one of the strongest aspects of the RiskWise system. Users expressed confidence that their personal information and submitted reports are protected from unauthorized access. The authentication and login processes were perceived as secure, contributing to overall user trust in the system. These results indicate that RiskWise effectively implements security measures that safeguard sensitive data and support safe system usage.

- **Maintainability**

  Based on the evaluation by IT experts, the system demonstrates good maintainability. Respondents indicated that the system can be updated or modified with relative ease when requirements change. The code structure was found to support efficient debugging and future enhancements. This suggests that the system is well-organized and can accommodate updates and improvements over time without major difficulty.

- **Portability**

  The results show that RiskWise is portable and adaptable to different environments. Respondents agreed that the system works properly when accessed on different computers and devices. Additionally, the system was perceived as capable of being deployed in other campuses or institutional settings with minimal modifications. This indicates strong potential for scalability and future expansion.Functional Suitability.

4-Point Likert Scale

Scale	Verbal Interpretation
4	Strongly Agree
3	Agree
2	Disagree
1	Strongly Disagree

Interpretation of Results

To interpret the values obtained from *WM*, *WAM*, *TWAM*, and *OWAM*, the following scale will be used:

Scale Range	Verbal Interpretation	Description
3.26 – 4.00	Strongly Agree	The system fully satisfies the requirement or criterion.
2.51 – 3.25	Agree	The system satisfactorily meets the requirement or criterion.
1.76 – 2.50	Disagree	The system partially meets the requirement or criterion.
1.00 – 1.75	Strongly Disagree	The system does not meet the requirement or criterion at all.

This interpretation scale will guide the analysis of each item, ISO/IEC 25010 criteria, respondent group total, and the overall system evaluation.

Software Development Methodology

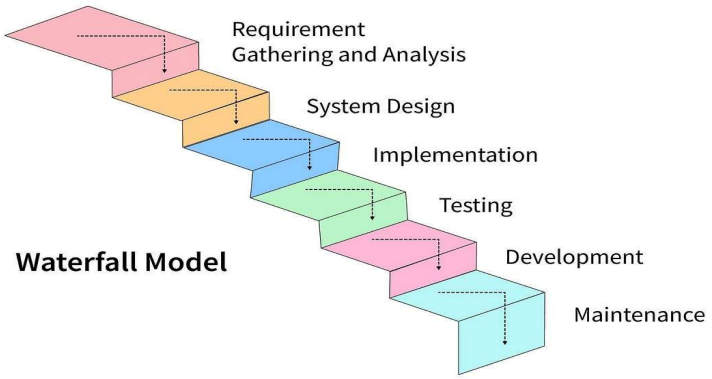


Figure 2: *Waterfall Methodology*

The study adopted the Waterfall Model as the software development methodology due to its structured and sequential nature, which is suitable for projects with clearly defined requirements and objectives. Each phase of the model was completed before proceeding to the next to ensure systematic development and proper documentation.

Requirements Phase

The requirements analysis phase served as the foundation of the entire system development process. During this phase, the researchers focused on identifying and documenting the functional and non-functional requirements of the RiskWise Hazard Reporting and Monitoring System. Data were gathered through interviews, informal discussions, and observations involving students, faculty members, non-teaching staff, and campus safety personnel of Pampanga State University. These stakeholders

provided insights into the existing hazard reporting process, common safety issues encountered on campus, and the limitations of the current reporting methods.

The researchers analyzed the collected information to determine critical system functionalities. These included the ability for users to submit hazard reports with detailed descriptions, photographic evidence, and geolocation data; the ability for administrators to view, classify, and manage reports; and the capability to send real-time notifications to users. Non-functional requirements such as system usability, data security, reliability, performance efficiency, and compatibility across different devices and browsers were also identified. The outcome of this phase was a comprehensive set of documented requirements that clearly defined the scope and objectives of the system and served as the basis for subsequent design and development activities.

Instead of using traditional Data Flow Diagrams (DFDs), the RiskWise system adopts a cloud-based, event-driven architecture using Firebase, a NoSQL database platform. Due to the nature of NoSQL databases and real-time cloud services, system processes are handled dynamically through direct interactions between the client application, backend services, and Firebase rather than through linear data flow structures typically represented in DFDs.

Firebase manages data storage, retrieval, synchronization, and real-time updates automatically through document-based collections. User actions such as hazard report submission, authentication, status updates, and notification triggering directly interact with Firebase services without the need for predefined process-level data

flow modeling. This architecture allows the system to support real-time updates, offline data synchronization, and scalable data handling across multiple users and devices.

In addition, Firebase integrates built-in authentication, cloud messaging, and database services, which reduces the need for explicit data flow modeling. Data is securely stored and synchronized across the system through Firebase's internal mechanisms, ensuring consistency and reliability without relying on traditional DFD representations. As a result, the system design focuses on system flowcharts and architectural descriptions rather than DFDs, as these provide a more accurate representation of the system's behavior and structure.

This approach aligns with modern web application development practices, particularly for Progressive Web Applications that rely on cloud-based, real-time databases. The use of Firebase as a NoSQL database enables flexible data structures, efficient handling of unstructured data, and seamless scalability, making traditional DFDs unnecessary for the RiskWise system.Instead of using traditional Data Flow Diagrams (DFDs), the RiskWise system adopts a cloud-based, event-driven architecture using Firebase, a NoSQL database platform. Due to the nature of NoSQL databases and real-time cloud services, system processes are handled dynamically through direct interactions between the client application, backend services, and Firebase rather than through linear data flow structures typically represented in DFDs.

Firebase manages data storage, retrieval, synchronization, and real-time updates automatically through document-based collections. User actions such as hazard report submission, authentication, status updates, and notification triggering directly

interact with Firebase services without the need for predefined process-level data flow modeling. This architecture allows the system to support real-time updates, offline data synchronization, and scalable data handling across multiple users and devices.

In addition, Firebase integrates built-in authentication, cloud messaging, and database services, which reduces the need for explicit data flow modeling. Data is securely stored and synchronized across the system through Firebase's internal mechanisms, ensuring consistency and reliability without relying on traditional DFD representations. As a result, the system design focuses on system flowcharts and architectural descriptions rather than DFDs, as these provide a more accurate representation of the system's behavior and structure.

This approach aligns with modern web application development practices, particularly for Progressive Web Applications that rely on cloud-based, real-time databases. The use of Firebase as a NoSQL database enables flexible data structures, efficient handling of unstructured data, and seamless scalability, making traditional DFDs unnecessary for the RiskWise system.

2. Testing Phase

The testing phase was conducted to verify that the system functions correctly, reliably, and in accordance with the specified requirements. Multiple testing strategies were employed to ensure system quality. Unit testing was performed on individ-

ual components to validate their functionality and detect errors at an early stage. Integration testing was carried out to ensure seamless interaction between the front-end interface, back-end services, and database components.

Usability testing was also conducted by allowing selected users to interact with the system under real-world conditions. This testing focused on evaluating ease of navigation, clarity of instructions, and overall user experience. Feedback gathered during this stage was analyzed and used to refine system features and interface elements. Performance testing was conducted to assess system responsiveness and loading times during typical usage. Identified defects and performance issues were resolved prior to deployment to ensure system stability and reliability.

3. Deployment Phase

The development phase combined various technologies that worked together to deliver the system's functionality. The front-end was built using React, chosen because its component-based approach allowed for cleaner and more predictable UI development. The backend logic was separated into two parts: a main API built with Node.js, and a microservice component (used mainly for internal handling and auxiliary operations) written in Flask. Data was stored and retrieved using Firestore, a cloud-hosted NoSQL database known for fast document-style storage, making it suitable for storing hazard reports that vary in content. Notifications, particularly alerts sent to users when administrators updated report statuses, were delivered through Firebase Cloud Messaging. Finally, the system was deployed using Salesforce,

which provided reliable hosting and stable runtime behavior.The development phase combined various technologies that worked together to deliver the system's functionality. The front-end was built using React, chosen because its component-based approach allowed for cleaner and more predictable UI development. The backend logic was separated into two parts: a main API built with Node.js, and a microservice component (used mainly for internal handling and auxiliary operations) written in Flask. Data was stored and retrieved using Firestore, a cloud-hosted NoSQL database known for fast document-style storage, making it suitable for storing hazard reports that vary in content. Notifications, particularly alerts sent to users when administrators updated report statuses, were delivered through Firebase Cloud Messaging. Finally, the system was deployed using Salesforce, which provided reliable hosting and stable runtime behavior.

4. Maintenance Phase

The maintenance stage guarantees the risk evaluation and management reporting system to work, and to be up to date with all the changes in user and institution necessities after its initial deployment. This stage calls for the examination of the system regularly for bugs, performance issues, or any other behavior that might be seen only when the system is accessed by more users under actual conditions. All the errors discovered are enfranchised to be gotten rid of through the installation of patches or by making minor adjustments of the code. The maintenance phase is also characterized by the surveillance of the system's interface for its quality of interaction, testing the correctness of the risk data stored in the cloud database and making

sure that the notifications and status updates are working normally. Updating the system whenever new hazard categories, user roles, or reporting guidelines are introduced by the institution is one of the maintenance activities. Regular improvements are also carried out on the basis of feedback from end-users and administrators, including refining interface elements, improving loading performance, and enhancing data security. The system may also be subject to improvements to facilitate new features or integrations as the institution's safety needs change. Maintenance phase, in general, is a very important stage in the life cycle of the RiskWise system as it keeps the system reliable and effective throughout its operational life.

CHAPTER IV

RESULT AND DISCUSION

Chapter IV presents the results and discussion of the system evaluation. This chapter focuses on analyzing the performance of the RiskWise system based on the ISO/IEC 25010 software quality characteristics. The discussion interprets the evaluation results in terms of functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability, and explains how the system addresses the problems identified in Chapter I.

System Flowcharts and Descriptions

To support a clearer understanding of the internal processes and functional logic of the RiskWise system, a set of system flowcharts was developed. These flowcharts illustrate the step-by-step processes involved in key system functionalities such as user hazard reporting, administrative monitoring, risk categorization and scoring, push notification handling, analytics generation, and offline data synchronization. The flowcharts serve as technical documentation that visually represents how data flows through the system and how decisions are processed internally.

However, since flowcharts are part of the system design and technical architecture rather than evaluation results, they are not discussed in detail within this chapter. All system flowcharts are presented in the Appendix section of this study to provide reference for readers who require a deeper technical understanding of the system's structure and operation. Chapter IV instead focuses on discussing the results

of the system evaluation based on the ISO/IEC 25010 software quality characteristics.System Flowcharts.

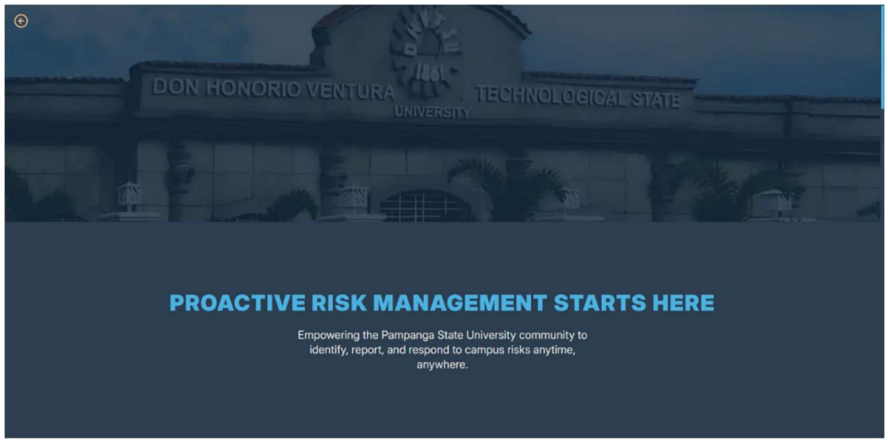
To support a clearer understanding of the internal processes and functional logic of the RiskWise system, a set of system flowcharts was developed. These flowcharts illustrate the step-by-step processes involved in key system functionalities such as user hazard reporting, administrative monitoring, risk categorization and scoring, push notification handling, analytics generation, and offline data synchronization. The flowcharts serve as technical documentation that visually represents how data flows through the system and how decisions are processed internally.

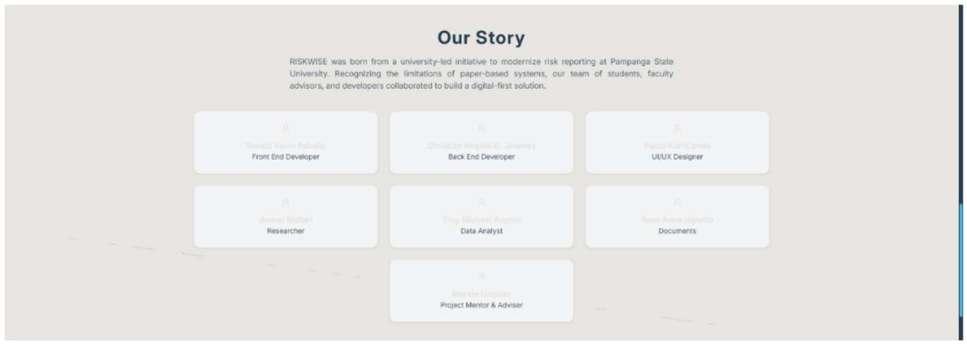
However, since flowcharts are part of the system design and technical architecture rather than evaluation results, they are not discussed in detail within this chapter. All system flowcharts are presented in the Appendix section of this study to provide reference for readers who require a deeper technical understanding of the system's structure and operation. Chapter IV instead focuses on discussing the results of the system evaluation based on the ISO/IEC 25010 software quality characteristics.

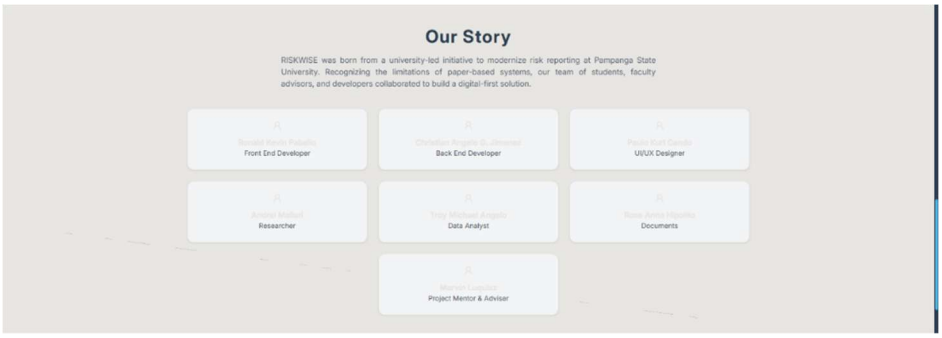
Sample Screenshots

Figure 3-7.

Front Page



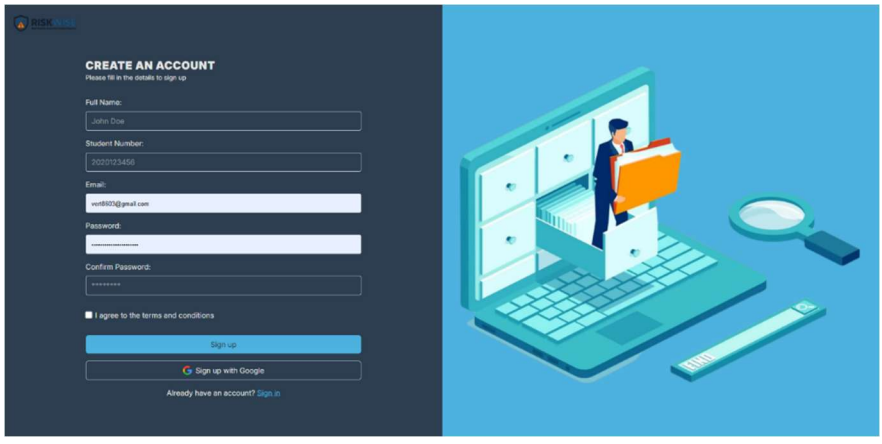




The front page serves as the system's entry point and is designed to be visually appealing and informative. It features a background image of the university, creating a sense of institutional identity. At the center, the system name "RISKWISE" is prominently displayed along with a short sentence A "Get Started" button is positioned below the sentence, guiding users toward the login or sign-up process. The top navigation menu includes links to About, Features, Download, and Contact, ensuring easy access to essential information.

Figure 8.

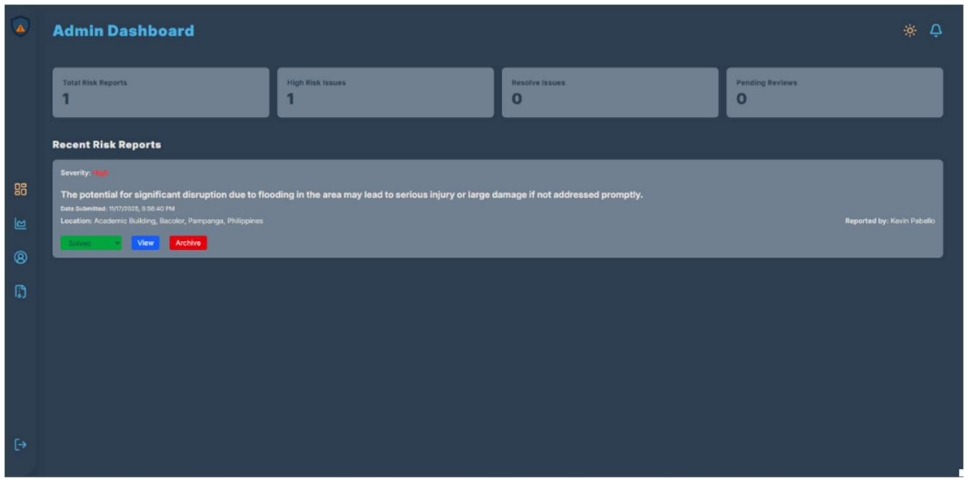
Login and Sign-Up Page



The login interface is designed for simplicity and security. Users can enter their email and password, with an option to show or hide the password for convenience. Below the login fields, an alternative Google Sign-In option is provided to streamline authentication. A “Forgot Password” button allows users to recover their accounts. For every sign-up and password reset request, the system sends a confirmation email to validate the user’s identity before granting access.

Figure 9.

Admin Dashboard

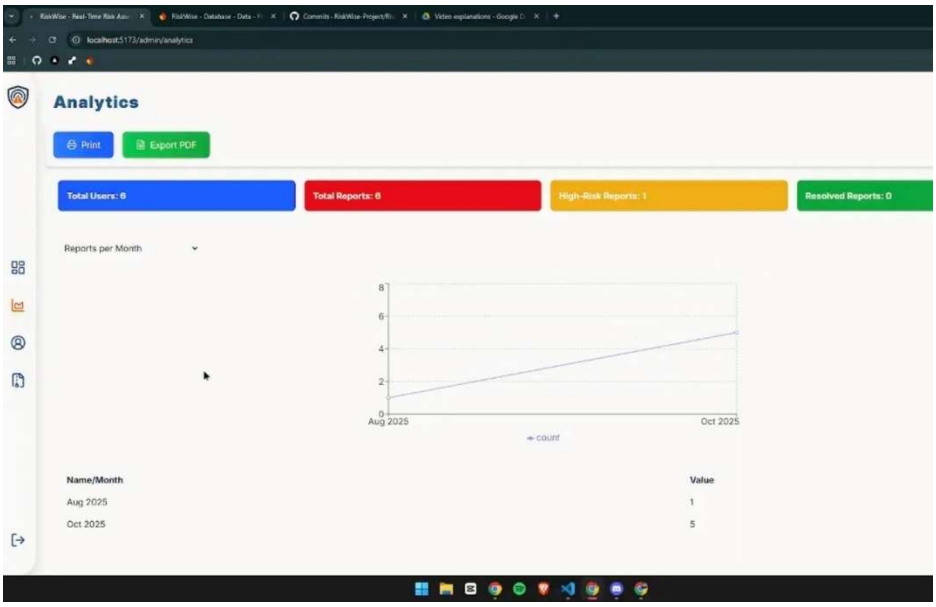


The admin dashboard displays a list of recent reports, each tagged with a severity level indicator. Reports with higher severity are automatically prioritized and appear at the top of the list, enabling administrators to address urgent issues promptly.

The dashboard also provides quick access to report details, including location, timestamp, and attached photos.

Figure 10.

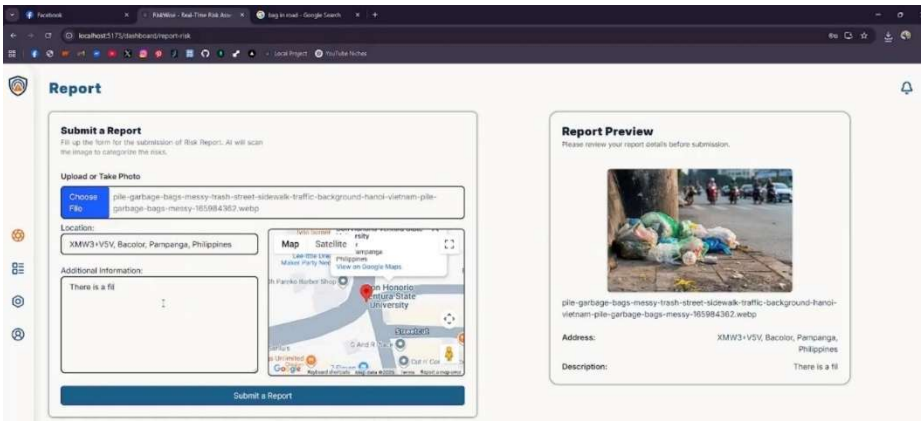
Analytics Page



The analytics interface offers administrators a visual representation of incident data through charts and graphs. These analytics summarize trends such as the frequency of hazards, their categories, and locations within the campus. By analyzing these patterns, administrators can make data-driven decisions to implement preventive measures and allocate resources effectively.

Figure 11.

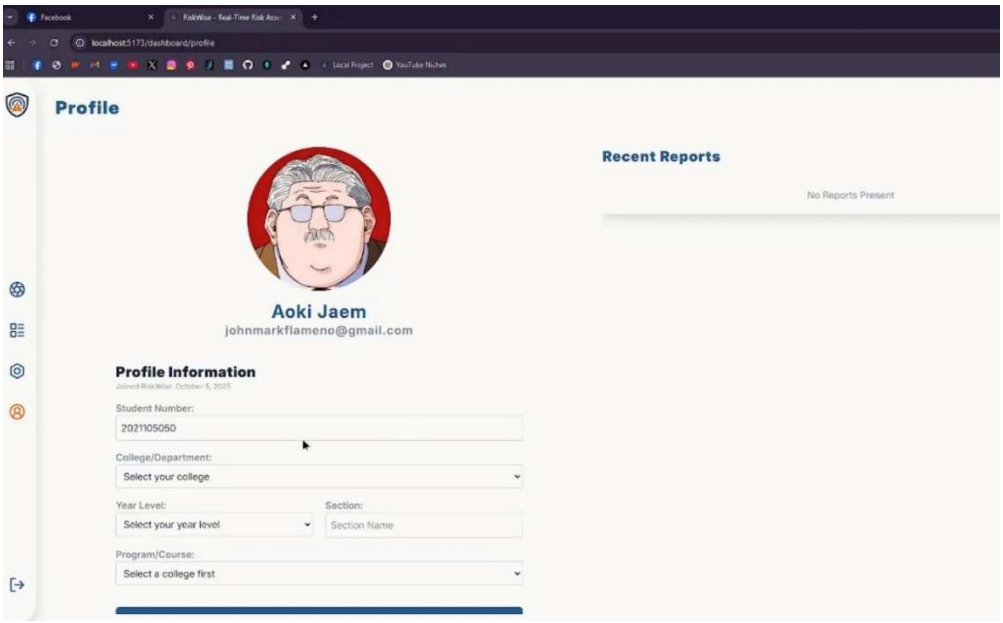
User Report Submission Page



The report submission page allows users to describe the hazard, attach photos, and pin the exact location of the incident on a map. This geolocation feature ensures precise reporting, while the photo attachment provides visual evidence for verification. The interface also includes an auto-timestamp to maintain accurate records.

Figure 12.

Profile View



The profile view enables users to manage their personal information and account settings. Users can update their password with email validation, upload a profile picture, and view their recent reports, ensuring transparency and personalization.

Figure 13.

Notification Panel

The notification panel keeps users informed about the status and actions taken on their submitted reports. Notifications include updates such as report acknowledgment, status changes, and administrative remarks, ensuring transparency and engagement.