Design and Implementation of a Web-Based Smart Device Management and Verification Platform

by

Sohag Akan
ID: CSE2301028017

Md. Atiq Wadud Lison
ID: CSE2301028054

Al-Amin
ID: CSE2301028136

Md. Mozammal Hossain
ID: CSE2301028146

Supervised by
Nayeem Ullah

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
September 2026

APPROVAL

This project titled **"Design and Implementation of a Web-Based Smart Device Management and Verification Platform"** has been submitted by Sohag Akan (CSE2301028017), Md. Atiq Wadud Lison (CSE2301028054), Al-Amin (CSE2301028136), and Md. Mozammal Hossain (CSE2301028146) to the Department of Computer Science and Engineering, Sonargaon University (SU), and has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Nayeem Ullah
Lecturer,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We hereby declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Nayeem Ullah, Lecturer** of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Nayeem Ullah)
Supervisor

Signature

Sohag Akan
ID: CSE2301028017

Md. Atiq Wadud Lison
ID: CSE2301028054

Al-Amin
ID: CSE2301028136

Md. Mozammal Hossain
ID: CSE2301028146

ABSTRACT

The increasing use of smartphones and laptops has raised concerns regarding lost, stolen, and second-hand smart devices, as users often face difficulties in determining whether a device has previously been reported as lost or stolen. To address this issue, this project presents TraceBack, a web-based smart device management and verification platform designed to support device reporting, searching, verification, and administrative management. The system provides separate facilities for reporting lost and found devices. Users can submit relevant information, including device brand, model, category, IMEI or Serial Number, location, contact details, description, images, and supporting proof documents. Each report is stored with a unique report code, verification information, and status. The IMEI/Serial Number Verification function enables users, particularly potential second-hand buyers, to check a device's reporting status before making a purchase decision. TraceBack also implements an administrator-approved matching workflow. Potential matches between lost and found reports remain under review until an administrator verifies the device information and ownership evidence. Following approval, the rightful owner can view the verified match information and claim instructions through the user dashboard. Sensitive documents and finder information remain restricted to authorized administrators. The frontend is developed using Next.js, while the backend is implemented using Laravel 12 and PHP, with SQLite as the database. API endpoints facilitate communication between the frontend and backend for report submission, device retrieval, verification, and status management, while matching and final approval are handled through backend logic and administrative review. Overall, TraceBack provides a centralized and structured platform for managing lost and found smart-device records while supporting controlled device recovery and safer verification of second-hand devices.

Keywords: Lost and Found System, Smart Device Management, IMEI Verification, Device Reporting, Second-Hand Device Verification, Web-Based Platform.

ACKNOWLEDGEMENT

At the very beginning, we would like to express our deepest gratitude to Almighty Allah for giving us the ability and strength to complete the task successfully within the scheduled time.

We are grateful for the kind guidance and supervision of **Nayeem Ullah**, Lecturer, Department of Computer Science and Engineering, Sonargaon University, whose valuable support and direction were essential to the successful completion of our project.

We would like to convey our special gratitude to **Prof. Bulbul Ahamed**, Head, Department of Computer Science and Engineering, Sonargaon University, for his kind support and valuable suggestions.

We are also thankful to all our teachers throughout our education for introducing us to the beauty of learning.

Finally, we express our deepest gratitude and love to our parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BTRC	Bangladesh Telecommunication Regulatory Commission
CORS	Cross-Origin Resource Sharing
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DB	Database
GD	General Diary
GSMA	GSM Association
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IMEI	International Mobile Equipment Identity
ITU	International Telecommunication Union
JS	JavaScript
JSON	JavaScript Object Notation
ML	Machine Learning
MVC	Model-View-Controller
NEIR	National Equipment Identity Register
NID	National Identity Card
NPM	Node Package Manager
OCR	Optical Character Recognition
ORM	Object-Relational Mapping
OWASP	Open Web Application Security Project
PHP	PHP: Hypertext Preprocessor
RBAC	Role-Based Access Control
REST	Representational State Transfer
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator

TABLE OF CONTENTS

Title		Page No.
DECLARATION		iii
ABSTRACT		iv
ACKNOWLEDGEMENT		v
LIST OF ABBREVIATIONS		vi
TABLE OF CONTENTS		vii-x
LIST OF TABLES		xi
LIST OF FIGURES		xii-xiii
CHAPTER 1		1 – 3
INTRODUCTION		
1.1	Background of the Project	1
1.2	Problem Statement	1-2
1.3	Objectives of the Project	2
1.4	Scope of the Project	2
1.5	Significance of the Project	3
1.6	Organization of the Project	3
CHAPTER 2		4 – 8
LITERATURE REVIEW		
2.1	Overview of Smart Device Management	4
2.2	Lost and Stolen Device Reporting	5
2.3	IMEI and Serial Number Verification	5-6
2.4	Found Device Reporting	6
2.5	Web-Based Device Management Systems	7
2.6	Limitations of Existing Approaches	7-8
2.7	Summary of Existing Works	8
CHAPTER 3		9 – 27
SYSTEM DESIGN AND METHODOLOGY		
3.1	Proposed TraceBack System Architecture	9
3.2	Frontend Architecture	10
3.2.1	User Authentication and Access Control	10-11

3.2.2 User Interface Design……... 11-12
3.2.3 Device Reporting Interface …….……..... 12-13
3.2.4 Device Search and Verification Interface …… 13-14
3.3 Backend Architecture …….. 14-15
3.3.1 API Design and Communication …….……. 15-16
3.3.2 Proof Document and File Management …….…….. 16-17
3.3.3 Admin Management and Verification …….……. 17-18
3.3.4 Security, Validation and Access Control …….…… 19
3.4 Database Design ……..... 19-20
3.4.1 Database Structure …….…….……. 20-21
3.4.2 Main Database Tables …….……..… 21-22
3.4.3 Entity Relationship Diagram (ERD) …….……... 22-23
3.5 Device Reporting Module …….…….…….. 23-24
3.6 Device Search and Verification Module …….. 24-25
3.7 Administrative Management …… 25-26
3.8 System Workflow …… 26-27

CHAPTER 4 28-37

IMPLEMENTATION AND CONFIGURATION

4.1 Development Environment …….. 28
4.2 Frontend Implementation ……. 28-29
4.3 User Authentication Implementation ……... 29-30
4.4 Backend Implementation ……. 30-31
4.5 Database Implementation ……...……...……..... 31-32
4.6 Device Report Model ……. 32-33
4.7 Device Report Controller ……..... 34-35
4.8 API Routes ……... 35-36
4.9 Admin Panel …… 36
4.10 File Upload and Proof Handling ……... 37

CHAPTER 5 38-45

TESTING AND VALIDATION

5.1 Testing Strategy ……... 38

5.2 Authentication Testing ……. 38-39

5.3 Health Check Testing ……... 39

5.4 Device Listing and Search Testing ……... 40

5.5 Report Submission Testing …….. 40-41

5.6 IMEI/Serial Verification Testing ……. 42-42

5.7 Status Update Testing ……..... 42

5.8 Admin Report Management Testing …… 43

5.9 File Upload Validation Testing ……..……... 43-44

5.10 Test Summary and Results ……..…….. 44-45

CHAPTER 6 46-56

RESULTS AND DISCUSSION

6.1 Functional Results …… 46

6.1.1 User Registration and Login Result ……...….. 46-47

6.1.2 Lost Device Reporting Result …….. 47

6.1.3 Lost Device Search Result ……... 48

6.1.4 Found Device Reporting Result ……... 48-49

6.1.5 Submitted Found Device Reporting Result ……..……... 49

6.2 Verification Results ……..…… 50

6.2.1 Device Verification Search Result …….. 50-51

6.2.2 Lost and Found Device Matching Result…….…… 51-52

6.3 Administrative Results ……. 52-53

6.3.1 Administrative Report Review ……. 53-54

6.3.2 Admin-Verified Match and Handover Information ……. 54-55

6.4 Usability and System Organization …….. 55

6.5 Limitations ……..…….. 56

CHAPTER 7 57

CONCLUSION AND FUTURE WORK

7.1 Conclusion ……..….. 57

7.2 Future Improvements …….……... 57

REFERENCES …….……... 58-60

LIST OF TABLES

Table No.	Title	Page No.
Table 2.1	Major Features of a Web-Based Device Management System	7
Table 2.2	Limitations of Existing Approaches	8
Table 3.1	Main Database Tables of the TraceBack System	21-22
Table 4.1	Development Environment and Technology Stack	28
Table 4.2	Main Frontend Pages of the TraceBack System	29
Table 4.3	Main Backend Components of the TraceBack System	31
Table 4.4	Main Database Migrations	32
Table 4.5	Main Methods of DeviceReportController	34
Table 4.6	Main API Routes of the TraceBack System	35-36
Table 4.7	Main Functions of the Admin Panel	36
Table 4.8	Proof Document Fields and Validation	37
Table 5.1	Testing Strategy of the TraceBack System	38
Table 5.2	Authentication Test Results	39
Table 5.3	Health Check Test	39
Table 5.4	Device Listing and Search Test Cases	40
Table 5.5	Report Submission Test Cases	41
Table 5.6	IMEI/Serial Verification Test Cases	41-42
Table 5.7	Status Update Test Cases	42
Table 5.8	Admin Report Management Test Cases	43
Table 5.9	File Upload Validation Test Cases	44
Table 5.10	Summary of System Testing	45
Table 6.1	Functional Results of the TraceBack System	46
Table 6.2	Verification Results of the TraceBack System	50
Table 6.3	Administrative Results of the TraceBack System	53

LIST OF FIGURES

Figure No.	Title	Page No.
Fig. 2.1	Overview of Smart Device Management	4
Fig. 2.2	Lost and Stolen Device Reporting	5
Fig. 2.3	IMEI and Serial Number Verification	6
Fig. 2.4	Found Device Reporting	6
Fig. 3.1	Proposed TraceBack System Architecture	9
Fig. 3.2	Frontend Architecture	10
Fig. 3.3	Account-Required Warning for Unauthorized Users	11
Fig. 3.4	User Interface Design of the TraceBack System	12
Fig. 3.5	Device Reporting Interface of the TraceBack System	13
Fig. 3.6	Device Search and Verification Interface of the TraceBack System	14
Fig. 3.7	Backend Architecture of the TraceBack System	15
Fig. 3.8	API Design and Communication Flow of the TraceBack System	16
Fig. 3.9	Proof Document and File Management Process of the TraceBack System	17
Fig. 3.10	Admin Management and Device Report Verification Process	18
Fig. 3.11	Security, Validation and Access-Control Process of the TraceBack System	19
Fig. 3.12	Database Structure of the TraceBack Lost and Found System	21
Fig. 3.13	Entity Relationship Diagram of the TraceBack Database	23
Fig. 3.14	Workflow of the Device Reporting Module	24
Fig. 3.15	Device Search and Verification Interface of the TraceBack System	25
Fig. 3.16	Administrative Management and Approval Workflow of the TraceBack System	26
Fig. 3.17	Overall System Workflow of the TraceBack Lost and Found System	27
Fig. 4.1	User Authentication Interfaces: (a) Account-Required Warning, (b) User Login Interface, and (c) User Registration Interface	30
Fig. 4.2	Technical Database Schema Architecture of the TraceBack System	32
Fig. 6.1	User Registration, Login, and Protected-Access Results	47
Fig. 6.2	Lost Device Reporting Result	47
Fig. 6.3	Lost Device Search Result	48

Figure No.	Title	Page No.
Fig. 6.4	Found Device Reporting Result	49
Fig. 6.5	Submitted Found Device Reporting Result	49
Fig. 6.6	Device Verification Search Results: (a) No Direct Match Found, and (b) Match Found in Database	51
Fig. 6.7	Lost and Found Device Matching Result	52
Fig. 6.8	Administrative Report Review Result	54
Fig. 6.9	Admin-Verified Match and Handover Information	55

CHAPTER 1
INTRODUCTION

1.1 Background of the Project

Smartphones and laptops have become essential digital devices in modern society and are widely used for communication, education, professional activities, financial transactions, entertainment, and information management. As the use of mobile and connected devices continues to increase, the loss, theft, and unauthorized resale of such devices have become important concerns for device owners and consumers. The International Telecommunication Union (ITU) has identified the need for measures to combat the theft and reuse of stolen mobile devices and has discussed framewo.1rks for protecting consumers from the risks associated with stolen devices [1]. One important mechanism for identifying mobile devices is the International Mobile Equipment Identity (IMEI). The GSMA explains that an IMEI is a unique device identifier for cellular devices and that its Device Check service can be used to determine whether a device has been reported as lost or stolen in the GSMA Device Registry [2]. This demonstrates the importance of device-identifier-based verification when assessing a device's status before use or resale [6], [7]. In Bangladesh, the National Equipment Identity Register (NEIR) provides an official mechanism for checking the status of mobile devices using their IMEI numbers. The Bangladesh Telecommunication Regulatory Commission (BTRC) states that users can check whether a mobile device is authentic, legally imported, and reported as lost, stolen, or cloned through the NEIR system [3]. However, the proposed TraceBack project titled "Design and Implementation of a Web-Based Smart Device Management and Verification Platform" focuses on a broader, user-oriented reporting and management approach. The platform supports both mobile phones and laptops. It provides facilities for reporting lost and found devices, storing device identifiers such as IMEI or Serial Number, submitting supporting information and proof documents, and searching device records before a second-hand transaction. The proposed TraceBack platform therefore aims to provide a centralized environment where device owners can report lost devices, finders can submit found-device reports, and potential buyers can perform a basic identifier-based check against the records maintained by the platform. The system is intended to improve awareness and transparency during second-hand device transactions and support the organized management of lost, stolen, and found smart-device information.

1.2 Problem Statement

The increasing use of mobile phones and laptops has created a growing need for an organized system to manage information about lost, stolen, and found devices [6]. When a device is lost or stolen, the owner may not have a centralized platform where the device details, identification number, supporting documents, and contact information can be properly recorded and managed. As a result, important information about the lost device may become difficult to maintain or share. Another major problem is associated with the purchase of second-hand mobile phones and laptops. Before purchasing a used device, buyers may not have a convenient way to check whether the device has previously been reported as lost or stolen. Although IMEI-based device checking services are available for mobile devices, there is still a need for a user-oriented platform that can maintain detailed

device reports and support both mobile phones and laptops using IMEI or Serial Number [2], [3], [16]. Furthermore, when someone finds a lost mobile phone or laptop, there may be no simple centralized method for submitting the found-device information so that it can be connected with existing device reports. The absence of organized reporting, searching, supporting-document management, and administrative control can make the recovery and verification process more difficult. Therefore, there is a need for a centralized web-based platform that can allow users to report lost and found devices, store relevant device information and supporting proof, search devices using IMEI or Serial Number, and manage reported records through an administrative interface. The proposed TraceBack project titled "Design and Implementation of a Web-Based Smart Device Management and Verification Platform" aims to address these problems by providing these functions within a single web-based system.

1.3 Objectives of the Project

The main objective of this project is to develop a web-based platform for managing lost, stolen, and found smart devices. The system is designed to support mobile phones, laptops, and smart-device accessories, allowing users to submit device information using IMEI or Serial Number. It provides a search mechanism for checking reported device records and helps second-hand buyers verify device information before purchasing. Users can also submit supporting documents and device images to assist in the verification process. The platform provides a facility for reporting found devices and allows administrators to manage device reports and their statuses. All device-related information is maintained in a centralized database for efficient management and retrieval. Overall, the system aims to improve awareness and transparency in second-hand smart-device transactions.

1.4 Scope of the Project

The scope of this project covers the development of a centralized web-based platform for managing and verifying information related to mobile phones, laptops, and smart-device accessories. The system is designed to support users in reporting lost devices, reporting found devices, and checking previously reported device information. The platform allows users to submit important information about a device, including device category, brand, model, IMEI or Serial Number, color, location, contact information, description, device images, and supporting documents. Users can also provide supporting evidence such as an original box or purchase proof, GD copy, warranty or voucher information, where applicable. A major part of the system is the device search and verification facility. Users, particularly potential second-hand buyers, can search using an IMEI or Serial Number to determine whether a corresponding device record exists in the platform. If a matching record is found, the system can display the relevant reported information to help the user make a more informed decision. The system also includes a Found Device Reporting feature. A person who finds a lost mobile phone, laptop, or supported accessory can submit information through the platform. An administrative interface is included for managing submitted reports, reviewing information, and updating device status. The project focuses on the management of records maintained within the platform's own database. It does not currently provide direct integration with external police, telecom operators, manufacturers, or government databases.

1.5 Significance of the Project

The proposed TraceBack project titled "Design and Implementation of a Web-Based Smart Device Management and Verification Platform" provides a centralized platform for managing lost, stolen, and found mobile phones, laptops, and smart-device accessories. It allows device owners to report lost devices with important information and supporting documents. The platform helps second-hand buyers verify a device using its IMEI or Serial Number before purchase [2], [3]. Users can also report found devices, which may help reconnect them with their rightful owners. The system supports storing device images, original box or purchase proof, GD copy, warranty or voucher information, and other relevant details. From a technical perspective, the project demonstrates the implementation of frontend, backend API, database management, authentication, device reporting, search and verification, file uploading, and administrative management. Overall, the platform promotes organized device management and safer second-hand device transactions.

1.6 Organization of the Project

This project report is organized into seven chapters, with each chapter focusing on a specific part of the design, development, implementation, and evaluation of the proposed TraceBack platform. Chapter 1 introduces the project, including the background, problem statement, objectives, scope, significance, and organization of the report. Chapter 2 discusses the literature review and related concepts, including existing approaches to smart-device management, lost and stolen device reporting, device verification, and second-hand device transactions. Chapter 3 presents the proposed TraceBack system design and methodology, including the system architecture, frontend architecture, backend architecture using Laravel, database design, device-reporting workflow, IMEI/Serial Number verification, API design, proof document management, and administrative functions. Chapter 4 explains the implementation of the TraceBack platform, including the development environment, frontend, Laravel backend, database, DeviceReport model, controller, API routes, admin panel, file upload, proof handling, and sample data. Chapter 5 focuses on testing and validation of the developed system, including device reporting, device searching, verification, file uploading, status management, and administrative functions. Chapter 6 presents the results and discussion of the system in terms of functionality, verification, usability, performance, administrative features, and limitations. Finally, Chapter 7 concludes the project and discusses possible improvements and future development opportunities.

CHAPTER 2
LITERATURE REVIEW

2.1 Overview of Smart Device Management

Smart device management refers to the organized process of maintaining, reporting, identifying, searching, and verifying information related to smart devices such as mobile phones, laptops, and accessories. Devices can be identified using identifiers such as IMEI for cellular mobile devices and Serial Number for laptops and other electronic devices [2], [7]. A smart device management system provides a centralized platform for lost-device reporting, found-device reporting, device searching, verification, document submission, and administrative management. The proposed TraceBack platform follows this concept by allowing users to report lost or found devices with identification numbers, descriptions, images, and supporting documents. Potential second-hand buyers can search using an IMEI or Serial Number to check whether a corresponding record exists in the database. Therefore, the proposed TraceBack system focuses on organized device-record management and provides a simple mechanism for reporting, searching, and verifying smart-device information, helping users make more informed decisions during second-hand device transactions. The overall concept of smart device management and verification, including user interaction, administrative control, database storage, and core system features, is illustrated in Fig. 2.1.

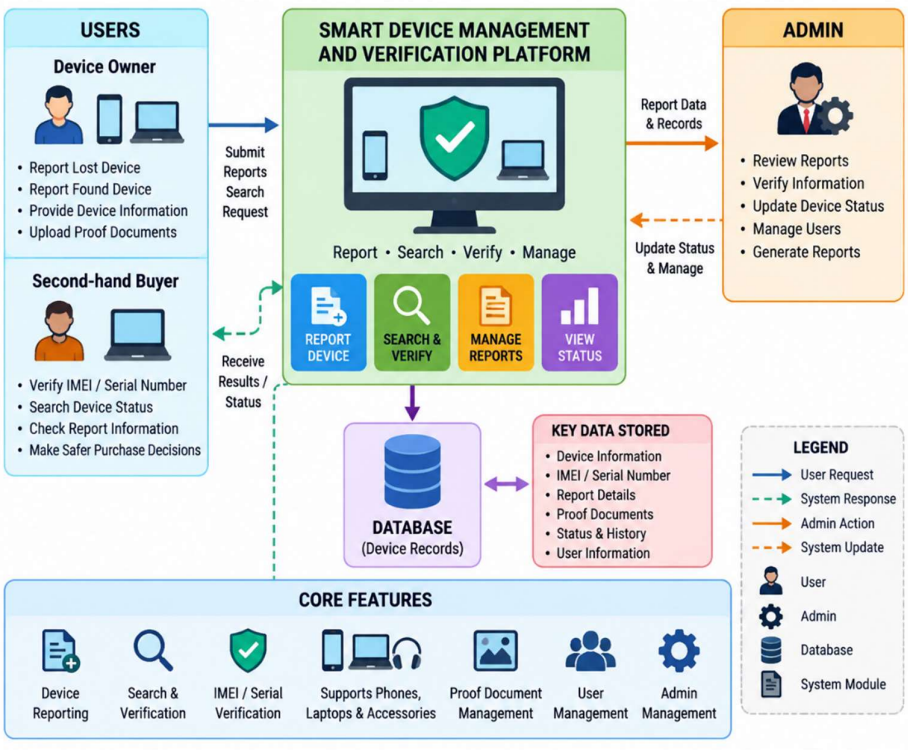


Fig. 2.1 Overview of Smart Device Management

2.2 Lost and Stolen Device Reporting

Lost and stolen device reporting is an important part of smart device management [6]. When a mobile phone, laptop, or accessory is lost or stolen, the owner can record important information about the device to help identify and manage the report. The proposed TraceBack platform allows users to submit details such as device type, brand, model, IMEI or Serial Number, device images, original box or purchase proof, GD copy, warranty or voucher information, location, and comments. The submitted information is stored in the centralized database for future searching and verification. This feature also helps potential buyers check whether a second-hand device has previously been reported as lost or stolen [2], [6]. If the submitted IMEI or Serial Number matches an existing report, the system can display the relevant status and help the buyer take appropriate caution before purchasing the device [2], [3]. The overall workflow of lost and stolen device reporting, including user submission, device information collection, proof document upload, validation, database storage, and report management, is illustrated in Fig. 2.2.

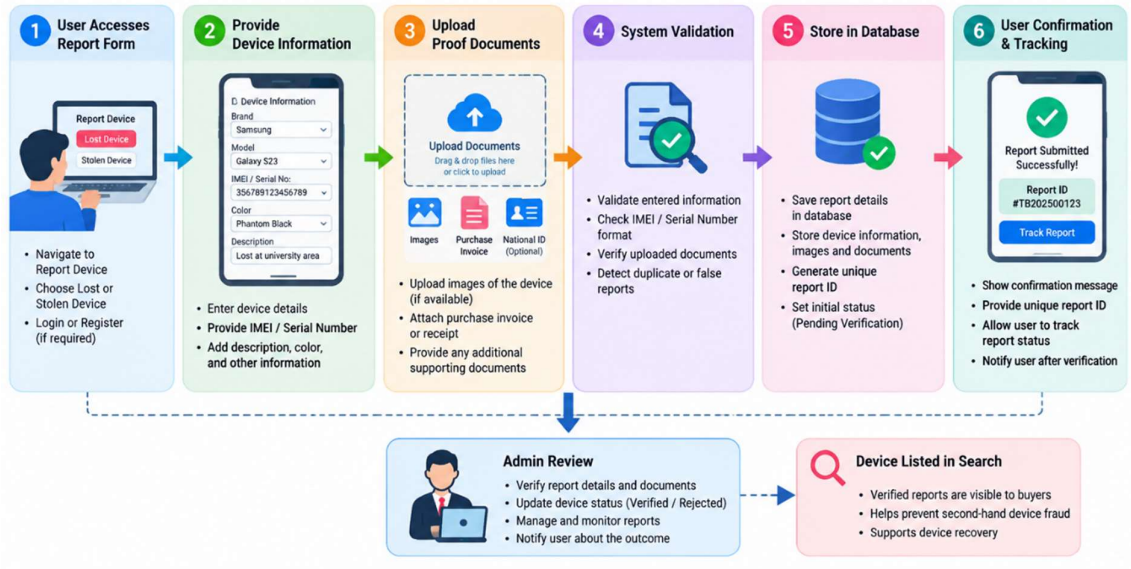


Fig. 2.2 Lost and Stolen Device Reporting

2.3 IMEI and Serial Number Verification

IMEI and serial numbers are important identifiers used to identify smart devices. The International Mobile Equipment Identity (IMEI) is mainly used for mobile phones and other cellular devices, whereas serial numbers are commonly used to identify laptops and other electronic devices. The proposed TraceBack platform provides a search and verification facility using an IMEI or serial number. When a user enters an identifier, the system searches the database for a matching device record. If a matching record is found, the available device information and reported status can be displayed to the user. If no matching record is found, the system indicates that no corresponding report exists in the platform database. This feature is particularly useful for second-hand device buyers because they can check the available record of a mobile phone or laptop before purchasing it [2], [3]. IMEI-based device-status checking is also used by existing services such as GSMA Device Check and Bangladesh's NEIR system [2], [3]. The IMEI and serial-number

verification process, including user input, identifier submission, database searching, record matching, and the display of verification results, is illustrated in Fig. 2.3.

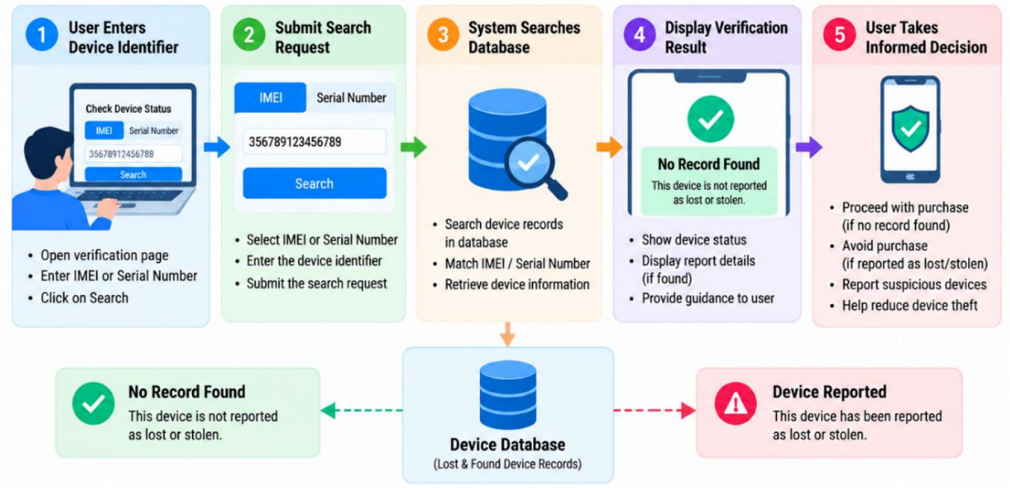


Fig. 2.3 IMEI and Serial Number Verification

2.4 Found Device Reporting

The Found Device Reporting feature allows a person who finds a lost mobile phone, laptop, or smart-device accessory to submit information about the device through the proposed TraceBack platform. The finder can provide details such as device type, brand, model, IMEI or Serial Number, found location, device images, description, and comments. The submitted information is stored in the platform database and can be managed by the administrator. This feature provides a structured way to record found devices and may help connect a found device with an existing lost-device report. It can therefore support the recovery process and encourage users to return lost devices to their rightful owners. The workflow of found device reporting, including user submission, device information collection, evidence upload, database storage, and administrative verification, is illustrated in Fig. 2.4.

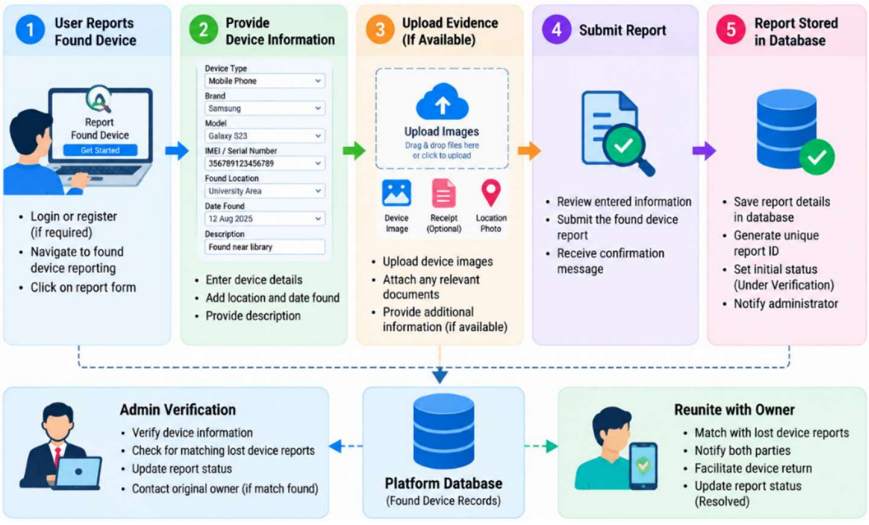


Fig. 2.4 Found Device Reporting

2.5 Web-Based Device Management Systems

Web-based device management systems provide a centralized environment for storing, managing, searching, and accessing device-related information [5]. Such systems can support functions such as user management, device reporting, database management, document and image uploading, device searching, verification, and administrative control. The proposed TraceBack platform follows a web-based approach to manage mobile phones, laptops, and smart-device accessories. Users can report lost or found devices, upload supporting information, and search for devices using their IMEI or Serial Number. The system stores the submitted information in a centralized database for future search and verification. Since the platform includes file and document uploading, proper validation and access control are important for secure web-based operation. OWASP recommends validating uploaded files, restricting permitted file types and sizes, and allowing uploads only for authorized users [4]. The major system features are summarized in Table 2.1.

Feature	**Purpose**
User Management	Manage registered users
Device Reporting	Report lost, stolen, or found devices.
Device Search	Search using IMEI or Serial Number
Verification	Check available device records.
Document Management	Store images and supporting documents
Database Management	Maintain device and report records.
Administration	Manage users and submitted reports.
Admin-Approved Matching	Controls lost/found matching, verification, and handover disclosure

Table 2.1 Major Features of a Web-Based Device Management System

2.6 Limitations of Existing Approaches

Existing device-verification approaches provide useful services for identifying mobile devices and checking their reported status. Services such as GSMA Device Check and Bangladesh NEIR mainly focus on IMEI-based mobile-device verification [2], [3]. However, the reviewed approaches do not describe an integrated user platform combining lost-device reporting, found-device reporting, laptop and accessory support, proof-document management, and administrator-approved matching. Therefore, TraceBack is proposed as a centralized web-based platform that combines these functions with IMEI/Serial Number searching and administrative control. The identified limitations and corresponding TraceBack features are summarized in Table 2.2.

Area	Existing Approaches	TraceBack Implementation	Reference(s)
Mobile Device Verification	Mainly IMEI-based	IMEI-based report search and status checking	[2], [3], [6], [7], [16]
Laptop Support	Not covered by IMEI-focused services	Serial-number-based reporting and searching	[2], [3]
Accessories Support	Not covered by reviewed services	Reporting through device details and identifiers	[2], [3]
Lost Reporting	Mainly mobile-theft and registry-based	Lost/stolen reports with ownership proof	[1], [2], [3], [6], [16]
Found Reporting	Integrated facility not described	Found-device reporting with location and details	[2], [3]
Document Management	Proof-upload facility not described	Secure storage of supporting documents	[2], [3]
IMEI/Serial Search	Mainly IMEI checking	IMEI, serial number, and report-code search	[2], [3], [7]
Centralized Platform	Mainly identification and status checking	Reporting, searching, verification, and dashboard	[1], [2], [3], [16]
Admin Management	Match-approval process not described	Report review, match approval, and status control	[1], [2], [3], [16]

Table 2.2 Limitations of Existing Approaches

2.7 Summary of Existing Works

Existing approaches show that device identification and verification can help users determine the status of mobile devices. GSMA Device Check allows device identifiers such as IMEI to be checked against reported lost or stolen records, while the GSMA Device Registry supports the sharing of device-status information [2]. The review also shows that existing approaches mainly focus on mobile-device identification and verification. The proposed TraceBack platform extends this concept by combining lost and stolen device reporting, found-device reporting, IMEI/Serial Number searching, supporting document management, and administrative control for mobile phones, laptops, and smart-device accessories. Overall, the existing approaches provide a useful foundation for device verification, while the proposed TraceBack platform aims to provide a more centralized and user-oriented solution for managing and verifying smart-device information [1], [2], [3].

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1 Proposed TraceBack System Architecture

The proposed TraceBack project titled "Design and Implementation of a Web-Based Smart Device Management and Verification Platform" is structured as a web-based system consisting of several interconnected components [8], [19], [20]. The architecture mainly includes the user interface, backend application, database, file-storage, and administration layers. Users can access the platform through the web interface to register, log in, report lost or stolen devices, report found devices, and search for devices using an IMEI or serial number. The frontend sends user requests to the backend, where the application processes the requests and communicates with the database. The backend manages user authentication, device reports, search and verification, file uploads, and administrative operations. Device information and report data are stored in the database, whereas supporting files such as original-box photographs, GD copies, warranty or voucher documents, and device images are managed through the file-management system. When a buyer searches for an IMEI or serial number, the backend checks the database for a matching record. If a record is found, the relevant device status or available information is returned to the user. Administrators can also review and manage submitted reports and user information. The overall architecture of the proposed TraceBack system, including the frontend interface, backend application, database, file storage, and administration layers, is illustrated in Fig. 3.1.

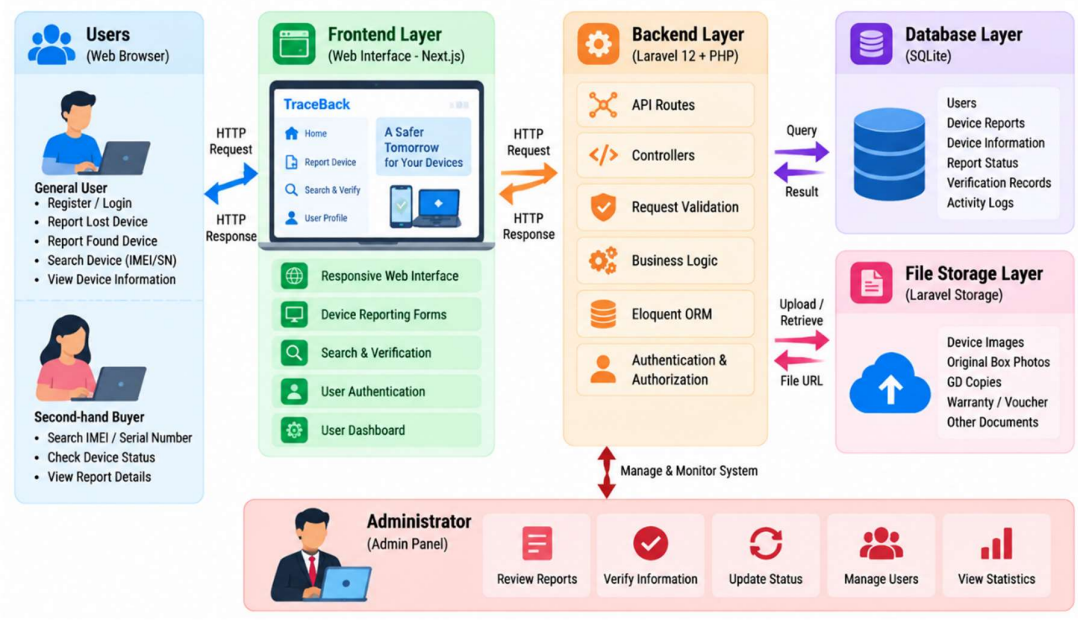


Fig. 3.1 Proposed TraceBack System Architecture

3.2 Frontend Architecture

The frontend architecture represents the user-facing part of the proposed TraceBack platform. It provides the interface through which users can register, log in, report lost or stolen devices, report found devices, and search for devices using an IMEI or serial number. Modern web applications commonly use client-side technologies to provide interactive user interfaces and communicate with server-side components. Research on web application architecture discusses different client-side application models and the role of browser-based technologies in modern web applications [5]. In the proposed TraceBack system, the frontend receives user input, sends requests to the backend through API communication, and displays the returned results. The interface is designed to provide separate functions for device reporting, device search and verification, user authentication, and administrative activities [21], [22]. The frontend architecture of the proposed TraceBack system, including the user-interface layer, application components, routing, API communication, and interaction with backend services, is illustrated in Fig. 3.2.

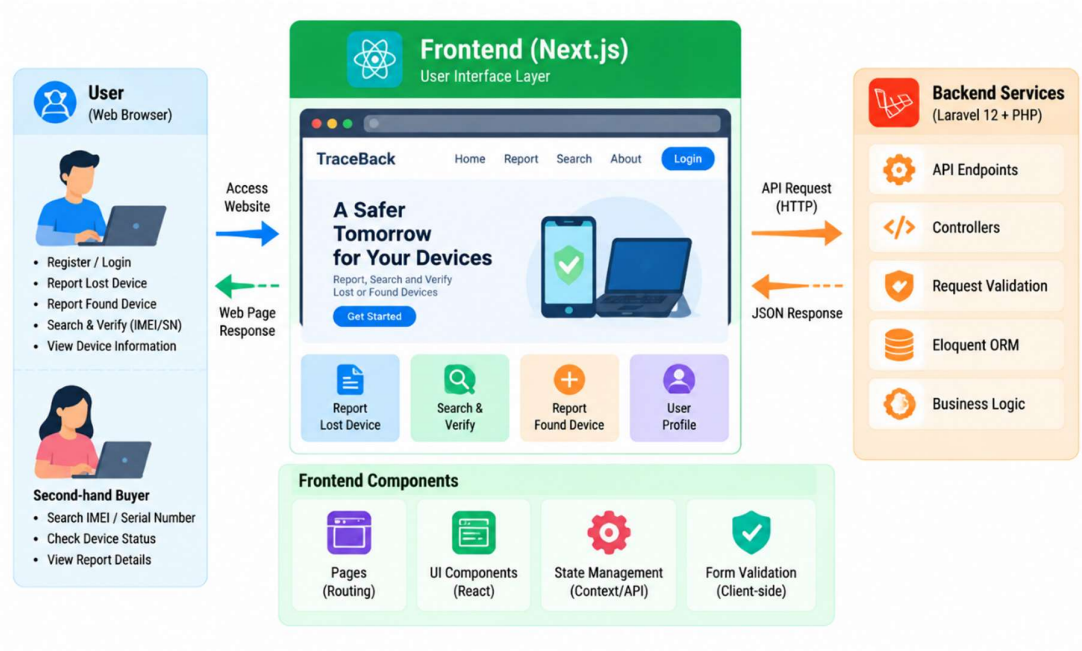


Fig. 3.2 Frontend Architecture

3.2.1 User Authentication and Access Control

The TraceBack platform implements user authentication and access control to protect device-reporting functions and reduce unauthorized or fraudulent submissions. A user must register and log in before submitting a lost- or found-device report or accessing account-specific information. If an unauthenticated user attempts to open a protected reporting page, the system displays a **Citizen Account Required** warning and provides options to sign in or create a new account. After successful authentication, the user is granted access to the reporting form and personal dashboard. This mechanism ensures that submitted reports are associated with authenticated users while administrative

functions remain restricted to authorized administrators. The account-required warning interface is shown in Fig. 3.3.

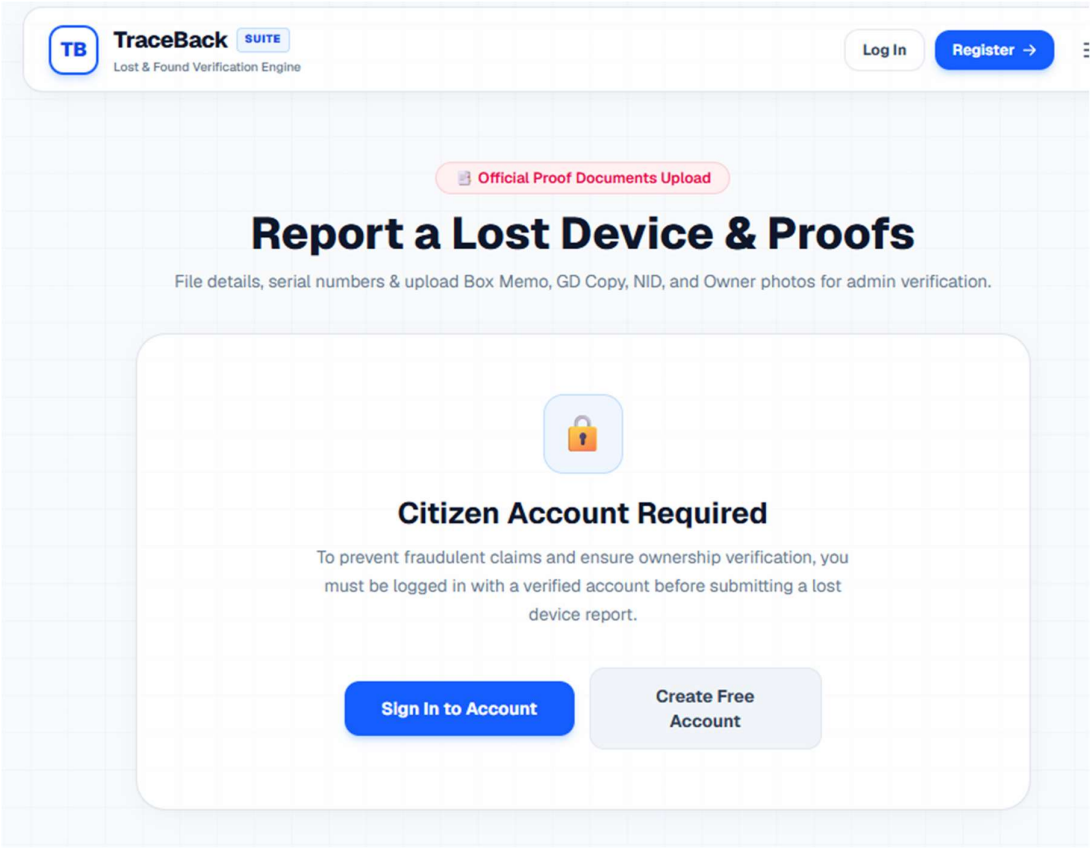


Fig. 3.3 Account-Required Warning for Unauthorized Users

3.2.2 User Interface Design

The User Interface (UI) of the proposed TraceBack platform is designed to provide a simple, clear, and user-friendly environment for different types of users, including device owners, finders, and buyers. The interface provides easy navigation to important functions such as reporting lost devices, reporting found devices, searching devices, and verifying device ownership. The home page provides an overview of the platform along with options for reporting lost devices and found devices. It also presents the verification process through a simple three-step interface, including device reporting or searching, uploading document proofs, and verified handover. The platform also includes a live verification catalog where users can search and view reported devices. Device information such as device type, brand, model, protected IMEI or serial number, color, location, and report status is displayed in an organized manner. For privacy and security, IMEI and serial numbers are masked in the public interface [21], [22], [23]. The user interface design of the TraceBack system, including the main pages, navigation structure, and user interaction components, is shown in Fig. 3.4.

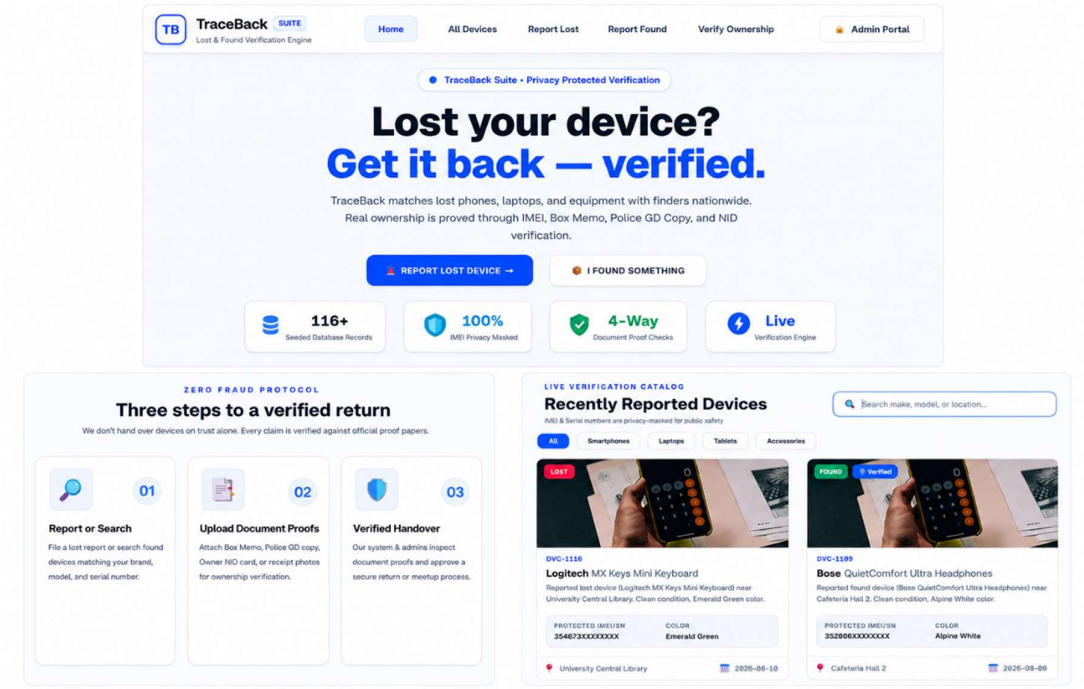


Fig. 3.4 User Interface Design of the TraceBack System

3.2.3 Device Reporting Interface

The Device Reporting Interface is one of the main components of the frontend of the proposed TraceBack platform. It provides users with a structured and user-friendly form for submitting information about lost or found devices. The interface allows users to select the device category and report type, where the report type can be either Lost or Found. Users are required to provide essential device information such as brand, model, IMEI or Serial Number, and location. Additional information, including device color, description, and contact information, can also be provided. These input fields correspond to the validation requirements implemented in the Laravel backend. For lost-device reports, the interface provides facilities for uploading supporting ownership documents, including Box/Memo Proof, GD Copy Proof, NID Proof, and Owner Picture Proof. For found-device reports, users can upload photographs of the found device or item. The uploaded documents and images are submitted along with the device report and processed by the backend for secure storage and verification. The backend validates the submitted information and uploaded files before storing the report. Supported proof files are checked according to their permitted formats and size limitations. After successful submission, the frontend sends the report information to the Laravel API. The backend validates the data, generates a unique device report code in the format DVC-XXXX, records the reporting date, and initially sets the report status to pending with the verification value set to false. The overall device reporting process can be summarized as: Select Report Type → Enter Device Information → Enter Location and Description → Upload Proof Documents → Submit Report → Backend Validation → Store Report → Generate Report Code → Set Initial

Status This interface provides a simple and organized mechanism for reporting lost and found devices while maintaining the required information for subsequent device searching, verification, recovery, and administrative management. The device reporting interface, including report type selection, device information input, proof document upload, and submission workflow, is illustrated in Fig. 3.5.

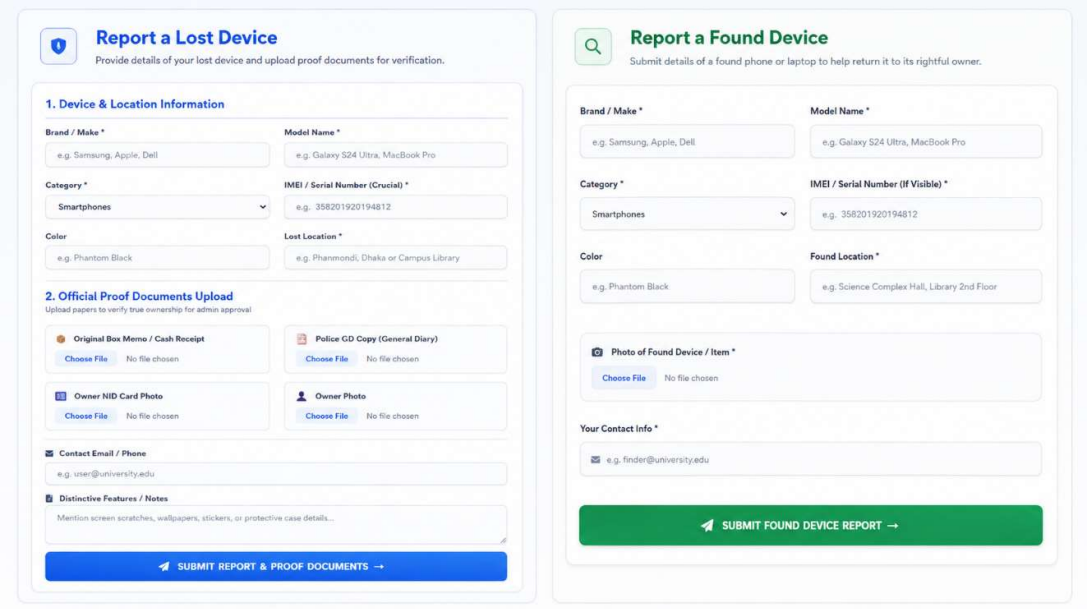


Fig. 3.5 Device Reporting Interface of the TraceBack System

3.2.4 Device Search and Verification Interface

The Device Search and Verification Interface is an important component of the frontend of the proposed TraceBack platform. It allows users, particularly second-hand buyers and device owners, to search for a device and check its reported status using an IMEI or Serial Number [2], [3]. The interface is designed to provide a simple and secure method for verifying whether a device has previously been reported as lost, stolen, or found. The search interface provides an input field where users can enter the IMEI or Serial Number of a device. After submitting the search request, the frontend sends the entered information to the Laravel backend through an API request. The backend processes the request and checks the database for a matching device record. If a matching record is found, the system returns the available device information and current report status to the frontend. For privacy and security, sensitive information such as the complete IMEI or Serial Number is displayed in a masked format rather than exposing the full identifier. The search result may include information such as device type, brand, model, color, location, report status, and verification status depending on the available record and user access permissions. If the device has been reported as lost or stolen, the interface clearly indicates the reported status so that potential buyers can take appropriate caution before purchasing the device. If no matching record is found, the system displays an appropriate message indicating that no report is currently available for the entered IMEI or Serial Number. The interface may also

guide users to verify ownership documents and available proof before completing a device transaction. The overall device search and verification process can be summarized as: Enter IMEI/Serial Number → Submit Search → Send API Request → Backend Validation → Database Search → Retrieve Device Record → Check Report Status → Display Verification Result This interface helps users make informed decisions before purchasing or accepting a device and provides an organized mechanism for checking device records within the proposed TraceBack platform. The device search and verification interface, including identifier input, database searching, record matching, and verification result display, is illustrated in Fig. 3.6.

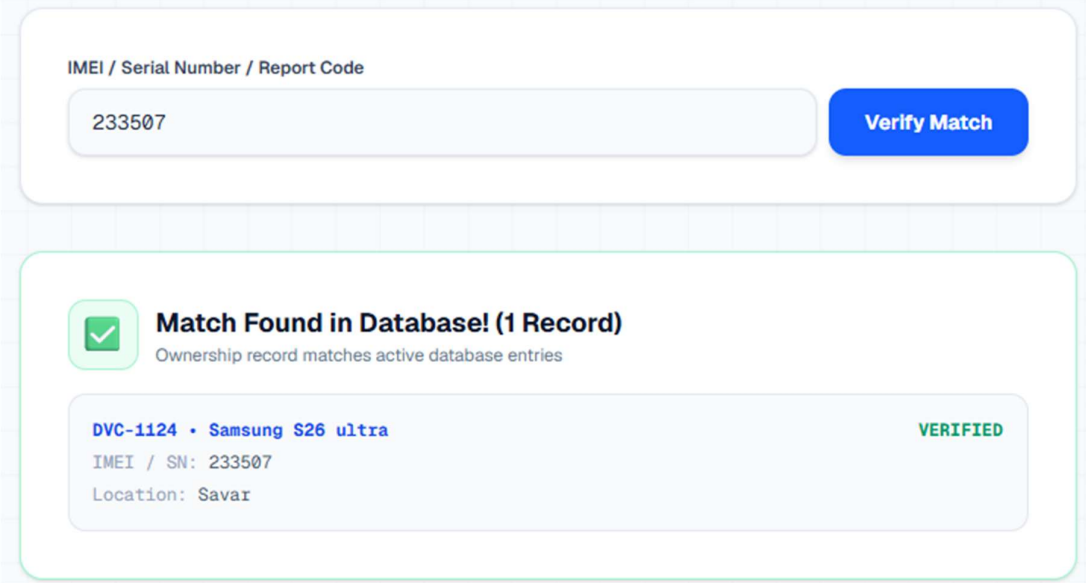


Fig. 3.6 Device Search and Verification Interface of the TraceBack System

3.3 Backend Architecture

The Backend Architecture represents the server-side component of the proposed TraceBack platform. It is responsible for processing user requests, validating submitted information, managing device reports, storing uploaded files, communicating with the database, and returning responses to the frontend. The backend is developed using Laravel 12 and PHP, following Laravel's structured application architecture. The frontend communicates with the backend through API endpoints. When a user submits a lost or found device report, the backend receives the device information and supporting files, validates the submitted data, and stores the valid information in the database. It also generates a unique report code, records the reporting date, and assigns an initial status to the report. The DeviceReportController handles the main backend operations, including retrieving device reports, applying category and report-type filters, searching records, creating new reports, verifying devices using an IMEI or serial number, updating report status, and deleting reports. The DeviceReport model is used to interact with the device-report records stored in the database through Laravel's Eloquent ORM. The backend also manages supporting files such as device images, purchase documents, GD copies, NID proofs, and owner

photographs through Laravel's file-storage system. File paths are stored in the database so that the uploaded evidence remains connected to the corresponding device report. For device verification, the backend receives an IMEI or serial number from the frontend and searches the platform database for a matching record. If a match is found, the available device information and current report status are returned. If no match is found, the system informs the user that no corresponding report currently exists in the platform database. Therefore, the backend architecture acts as the central processing layer between the frontend interface, database, and file-storage system [8]. It supports reliable data processing, device-record management, identifier-based searching, file handling, and administrative operations. However, the current backend operates only with the platform's own database and is not directly integrated with BTRC, NEIR, police, telecom operators, manufacturers, or the GSMA registry. The backend architecture of the proposed TraceBack system, including API communication, controllers, models, database interaction, file storage, and administrative operations, is illustrated in Fig. 3.7.

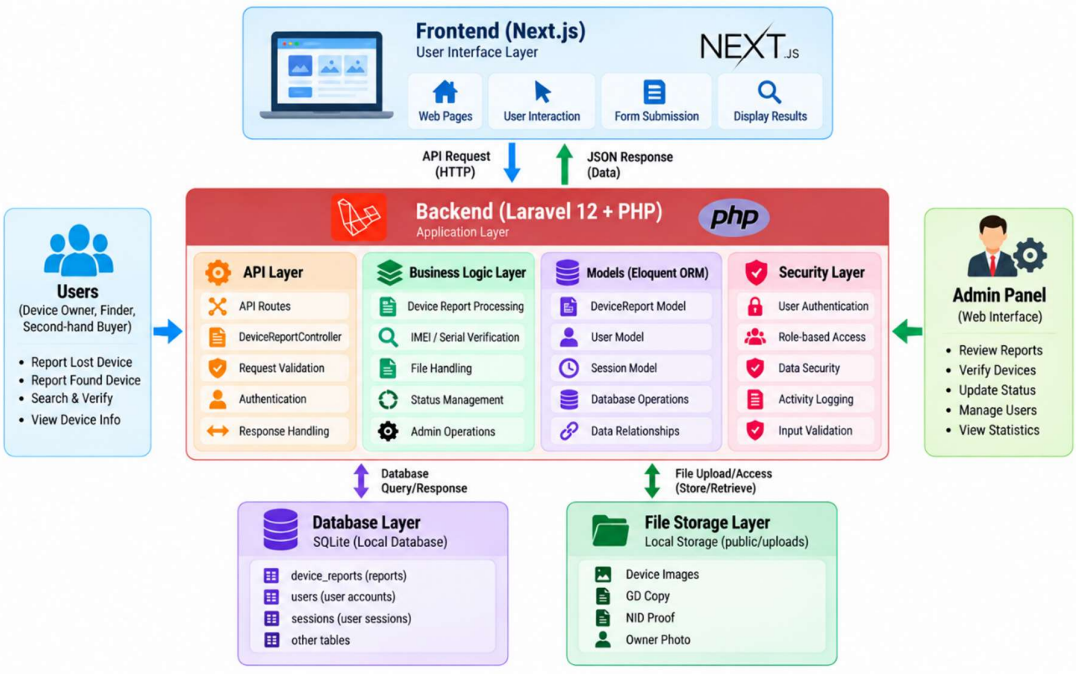


Fig. 3.7 Backend Architecture of the TraceBack System

3.3.1 API Design and Communication

The backend of the proposed TraceBack Lost and Found System uses a RESTful API developed with Laravel 12 and PHP [8]. The API acts as the communication layer between the Next.js frontend and the backend application. When a user performs an operation such as viewing or searching device reports, submitting a lost or found report, verifying an IMEI/serial number, or when an administrator updates the status of a report, the frontend sends an HTTP request to the corresponding Laravel API endpoint. The implemented API includes GET /devices for retrieving and searching device records, POST /reports for submitting new lost or found reports, POST /verify-imei for device verification, and PATCH /devices/{id}/status for updating report status. Incoming API requests are routed

to the appropriate method of the DeviceReportController. Before a new report is stored, Laravel validates required information such as brand, model, category, report type, IMEI or serial number, and location. Supporting proof files are also validated according to their permitted file types and maximum size. After successful validation, the system generates a unique device report code, records the reporting date, assigns the initial status as pending, and sets the verification state to false. The processed information is then stored through the DeviceReport model in the SQLite database. For verification, the backend accepts an IMEI/serial value and searches the stored device_reports records using the identifier or report code. The API returns the number of matching records together with the available matching information. Status updates are also processed through the controller, where valid states include pending, matched, verified, returned, and flagged. When a report is marked as verified or returned, its verification flag is set to true. After each operation, the Laravel backend returns a structured JSON response to the frontend containing the requested data, success information, matching results, or validation and error messages. This request–response mechanism provides organized communication between the frontend, backend controller, Eloquent model, SQLite database, and file-storage components of the proposed TraceBack system. The API design and communication flow of the proposed TraceBack system, including frontend requests, backend processing, controller operations, database interaction, and response generation, is illustrated in Fig. 3.8.

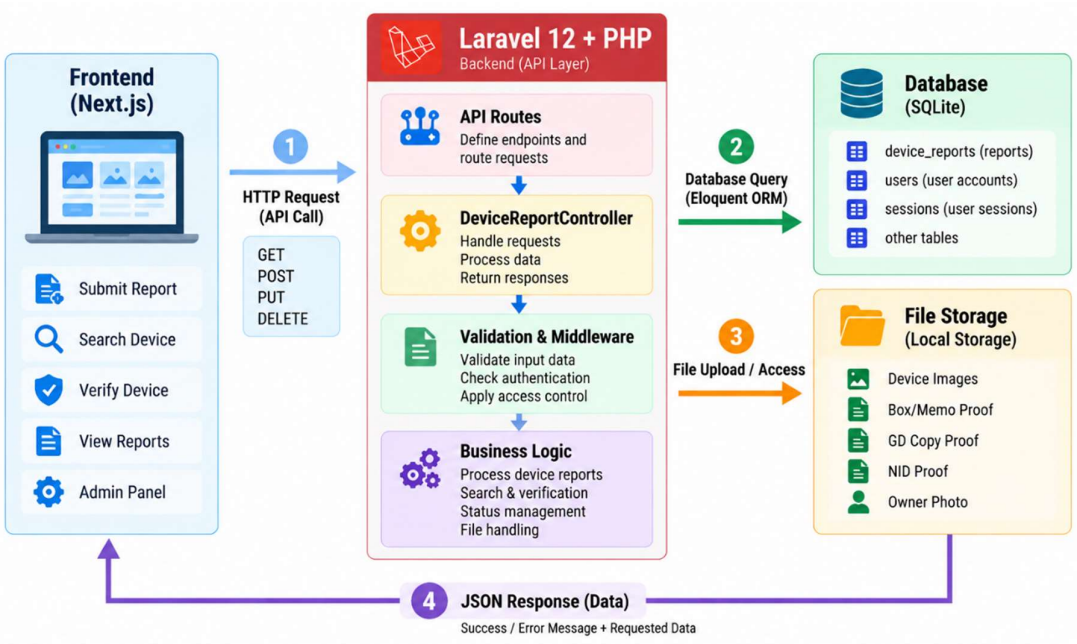


Fig. 3.8 API Design and Communication Flow of the TraceBack System

3.3.2 Proof Document and File Management

The proposed Web-Based Smart Device Management and Verification Platform supports the submission and management of proof documents with lost and found device reports. These documents provide additional evidence for device ownership and verification. Users can upload supporting files such as box or purchase memo proof, GD copy, NID proof, and owner photographs, which help administrators review reports and make verification

decisions. When a device report is submitted, the Laravel backend receives the uploaded files along with the report information. Before storage, the system validates the uploaded files based on format and size requirements [4]. The supported formats include JPG, JPEG, PNG, WEBP, GIF, BMP, PDF, and SVG, with a maximum size limit of 10 MB per file. Invalid files are rejected and an appropriate validation message is returned. After successful validation, files are stored using Laravel's public file-storage system. The actual files are stored separately from the SQLite database, while only their paths or URLs are stored in the device_reports table. The system maintains proof references through fields such as box_memo_proof, gd_copy_proof, nid_proof, and owner_pic_proof. This approach keeps the database organized and improves file management. During report verification, administrators can access the stored proof documents through their references and review the submitted evidence before updating the report status. Overall, the proof document management process combines file validation, secure storage, database references, and controlled retrieval to maintain supporting evidence efficiently. The proof document upload, validation, storage, and retrieval process of the TraceBack system is illustrated in Fig. 3.9.

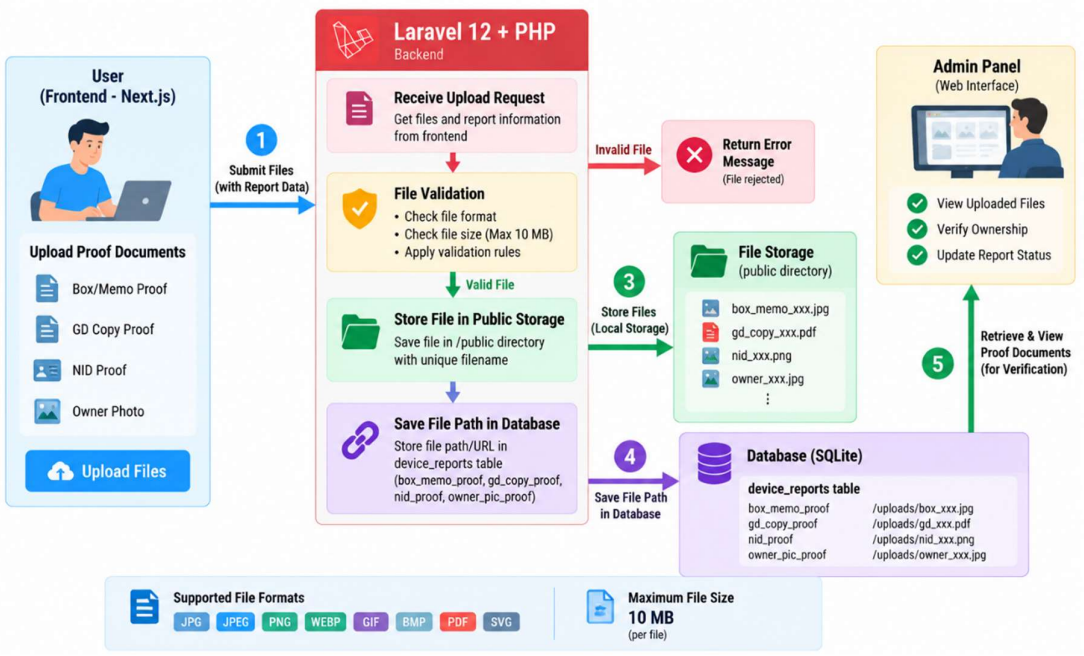


Fig. 3.9 Proof Document and File Management Process of the TraceBack System

3.3.3 Admin Management and Verification

The Admin Management and Verification Module provides a centralized interface for managing submitted device reports in the proposed TraceBack platform. It allows authorized administrators to review, verify, update, and manage lost and found device records while maintaining the accuracy and reliability of stored information. When a user submits a lost or found device report, the report details and related proof documents are stored in the SQLite database. The administrator can access the report list and examine important information, including device category, brand, model, IMEI/serial number, report type, location, description, device images, and supporting documents. During the

verification process, the administrator checks the submitted information and compares it with the uploaded proof documents to confirm validity. Based on the review, the administrator can verify the report, keep it pending for further investigation, or request additional information. The report status and verification flag are updated in the DeviceReport record through Laravel backend operations. The system also allows administrators to remove incorrect, duplicate, or unnecessary reports after proper review. Overall, this component improves system reliability by providing efficient report management, document verification, and status control through a single administrative platform. The implemented matching workflow follows an administrator-approval model. When a lost-device report and a found-device report contain the same IMEI or serial number, the system treats the pair only as a potential match. The match is not immediately disclosed to ordinary users. Both reports remain Under Review until an authorized administrator examines the identifiers, submitted ownership proofs, report details, and device condition, and then changes the relevant status to matched or verified. After administrative approval, the rightful owner's dashboard displays an Admin Verified Match banner and a "Matched By Admin" badge. A Match Info button opens a handover-details modal that presents the owner's reported device and the found device side by side, including the found location and date, finder notes or item condition, approved claim instructions, and verified helpdesk contact details. The owner can also print an official claim slip. Sensitive finder information and identity-document scans are restricted to authorized administrators and are never displayed to ordinary users [25], [26], [27]. The administrative management and device report verification workflow, including report review, information verification, status update, database modification, and user notification, is illustrated in Fig. 3.10.

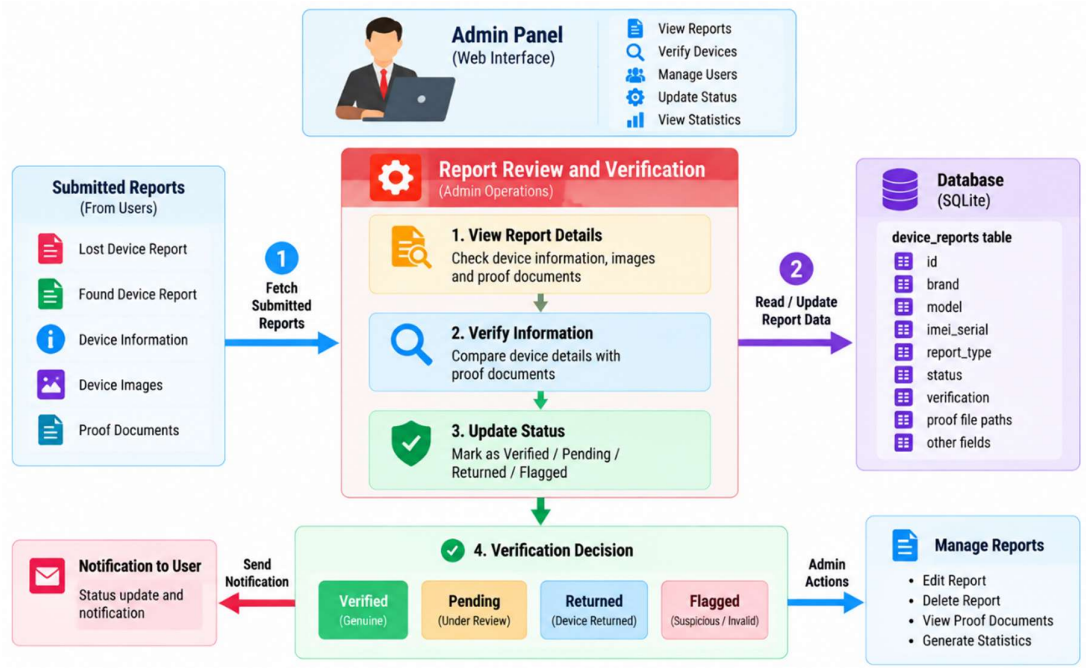


Fig. 3.10 Admin Management and Device Report Verification Process

3.3.4 Security, Validation and Access Control

Security and data validation are important for protecting device reports, proof documents, and administrative operations in the TraceBack system [4]. The system uses Laravel backend validation to verify all submitted information before storing or updating data. It checks required fields, device details, report type, IMEI/serial number, and uploaded files to ensure that only valid information is stored in the SQLite database. The backend rejects incomplete or incorrect requests and returns proper validation errors to prevent invalid records. Laravel's Eloquent ORM and secure query handling reduce the risk of SQL injection [14], while safe data output helps minimize security issues such as cross-site scripting [15]. Uploaded proof documents and device images are validated before storage, assigned unique file names, and stored separately, while only file paths are maintained in the database. Administrative functions such as report verification, status updates, and deletion are controlled through the admin interface and require backend validation before any changes are applied. Although the current system implements validation, secure file handling, and controlled operations, a complete role-based authentication and authorization system can be added in future improvements [24], [25], [26], [27], [28]. These security measures help maintain the confidentiality, integrity, and reliability of the TraceBack system. The security, validation, and access-control process of the TraceBack system, including input validation, file checking, secure processing, and controlled administrative operations, is illustrated in Fig. 3.11.

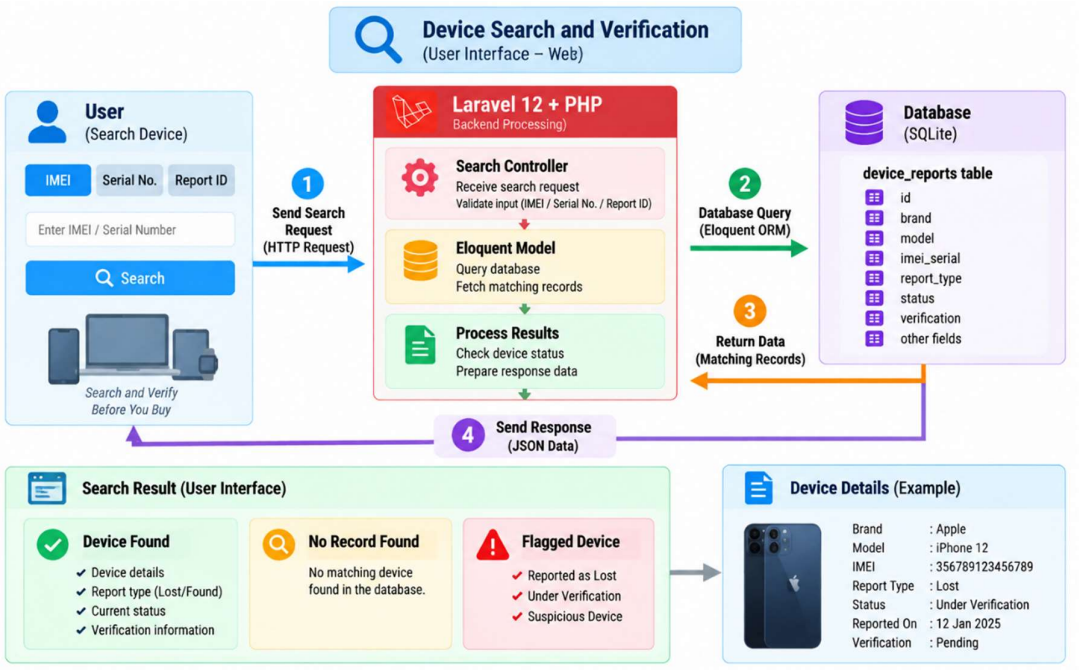


Fig. 3.11 Security, Validation and Access-Control Process of the TraceBack System

3.4 Database Design

The database is an essential component of the TraceBack Lost and Found System, as it is responsible for storing and managing information related to lost and found devices, users,

verification status, proof documents, and other supporting system data. The backend application uses SQLite as its default database management system, while Laravel migrations are used to create and maintain the database schema. The database is designed using a relational structure in which different types of information are stored in separate tables. The device_reports table serves as the main application table and stores information such as device brand, model, category, IMEI or serial number, report type, location, status, verification information, contact details, and proof-document references. The users table stores user and administrative account information, while additional Laravel tables are used for sessions, password-reset tokens, caching, and queue-related operations. Laravel's Eloquent ORM is used to interact with the database through models such as DeviceReport and User. This provides an organized way to perform database operations such as inserting new reports, searching for devices, updating report status, verifying records, and deleting information. The database also supports the storage of proof-document references, including box or purchase memo proof, GD copy, NID proof, and owner photograph. The actual uploaded files are stored separately in Laravel's public storage, while their corresponding URLs or paths are saved in the database. Important database constraints are also applied to improve data consistency and searching efficiency. For example, each device report contains a unique report code, while the IMEI or serial-number field is indexed to support faster searching and device verification. The report type is restricted to lost or found, and the status field is used to represent different stages such as pending, matched, verified, returned, and flagged. Overall, the database design provides a structured foundation for managing the major functions of the TraceBack system, including device reporting, searching, verification, proof management, user administration, and recovery-status tracking. The following subsections describe the database structure, main database tables, and Entity Relationship Diagram (ERD) in detail [32].

3.4.1 Database Structure

The TraceBack Lost and Found System uses a relational database structure to organize and manage the information required by the application. In the implemented system, SQLite is configured as the default database, while Laravel provides the application layer for communicating with and managing the stored data. The database structure is centered primarily on the device_reports table, which stores information about lost and found devices. This information includes the report code, device brand and model, category, report type, IMEI or serial number, color, location, reporting date, current status, verification state, description, contact information, and references to supporting proof documents. Other database tables support user accounts, authentication sessions, password recovery, caching, and background application processes. Laravel's Eloquent ORM connects the application logic to the database records. Models such as DeviceReport and User are used by the backend to retrieve, insert, update, and delete information. The database schema is maintained through Laravel migrations, which provide a structured approach for creating and modifying database tables during system development. Proof-document storage is separated from the structured database records. When documents such as a box or purchase memo, GD copy, NID proof, or owner photograph are uploaded, the files are stored through Laravel's file-storage mechanism, and their corresponding paths or URLs are recorded in the device_reports table. The database structure of the TraceBack

Lost and Found System, including the main application table, supporting tables, proof-document references, and data relationships, is illustrated in Fig. 3.12.

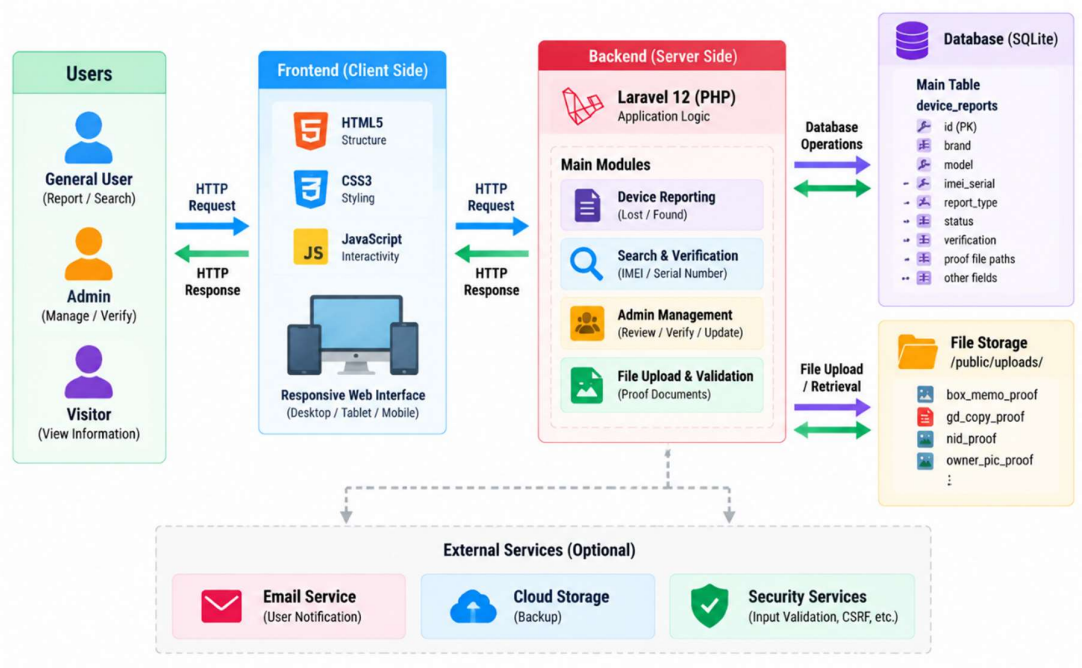


Fig. 3.12 Database Structure of the TraceBack Lost and Found System

3.4.2 Main Database Tables

The TraceBack database contains several tables that support device reporting, user management, authentication, and system operations. Among these, the device_reports table is the core application table because it stores all major information related to lost and found devices. The users, sessions, and password_reset_tokens tables support account and authentication-related functions. The main database tables are summarized in Table 3.1.

Table Name	Purpose	Important Fields
device_reports	Stores lost and found device reports, verification status, contact information, and proof-document references	id, code, brand, model, category, type, imei, color, location, date_reported, status, description, contact_info, verified, proof fields, created_at, updated_at
users	Stores user and administrator account information	id, name, email, email_verified_at, password, remember_token, created_at, updated_at

Table Name	Purpose	Important Fields
sessions	Stores information related to active user sessions	id, user_id, ip_address, user_agent, payload, last_activity
password_reset_tokens	Stores password recovery information	email, token, created_at

Table 3.1 Main Database Tables of the TraceBack System

The device_reports table is the most important table in the system. Each device report is identified by a primary key and a unique report code. It stores the device brand, model, category, report type, IMEI or serial number, color, location, reporting date, status, description, contact information, and verification state. The IMEI field is indexed to support efficient device searching and verification. The table also contains four fields for storing proof-document references: box_memo_proof, gd_copy_proof, nid_proof, and owner_pic_proof. These fields store the file path or URL of the corresponding verification documents. The users table stores account-related information such as name, email address, password, email verification status, and remember token. The email field is unique to prevent duplicate user accounts. The sessions table maintains session information such as the session ID, associated user ID, IP address, user-agent information, session payload, and last activity time. This information supports session-based authentication and user activity management. The password_reset_tokens table is used for password recovery. It stores the user's email address, generated reset token, and token creation time. In addition to these main tables, the Laravel backend also contains supporting tables such as cache, cache_locks, jobs, job_batches, and failed_jobs. These tables support internal framework operations including caching and background job processing rather than the primary lost-and-found workflow.

3.4.3 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) represents the logical structure of the main database entities used in the TraceBack Lost and Found System [9]. It illustrates the important tables, their major attributes, primary keys, and the relationships that are explicitly defined within the database structure. The main entities of the system are device_reports, users, sessions, and password_reset_tokens. Among these, the device_reports table is the central application entity because it stores information related to lost and found devices, verification status, contact information, and proof-document references. The users table stores account-related information such as the user's name, email address, password, email verification status, and authentication-related data. The sessions table stores session information and contains a user_id field for associating session records with users. The password_reset_tokens table stores email-based password recovery information. The Entity Relationship Diagram (ERD) of the TraceBack Lost and Found

System, including the main entities, attributes, primary keys, foreign keys, and relationships among database tables, is illustrated in Fig. 3.13.

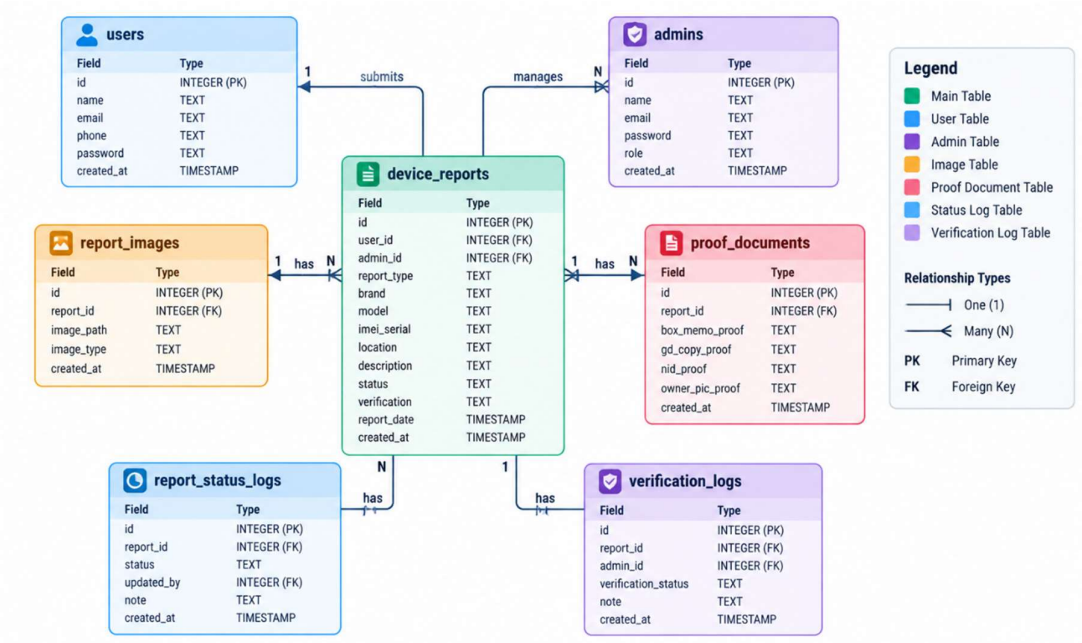


Fig. 3.13 Entity Relationship Diagram of the TraceBack Database

3.5 Device Reporting Module

The Device Reporting Module is one of the core functional components of the TraceBack Lost and Found System. It enables users to submit information about electronic devices that have been either lost or found. The module collects the necessary device information, validates the submitted data, processes supporting proof documents, and stores the complete report in the system database. A device report contains important information such as the device brand, model, category, report type, IMEI or serial number, color, location, description, and contact information. The report type identifies whether the submitted device is lost or found. The IMEI or serial number acts as an important identification attribute that can later be used for searching and verification. When a report is submitted from the frontend application, the information is sent to the Laravel backend through the device reporting API. The DeviceReportController receives the request and performs server-side validation before creating the report. Important information such as the device brand, model, category, report type, IMEI or serial number, and location must be provided before the report can be successfully stored. The module also supports the submission of verification documents. Users can provide a box or purchase memo, GD copy, NID proof, and owner photograph. Uploaded files are processed by the Laravel backend and stored in the public proof-storage directory, while the corresponding file URLs are maintained with the device report record. After successful validation, the system automatically generates a unique report code in the format DVC-XXXX and records the current reporting date. Every newly submitted report is initially assigned the status pending,

and its verification state is set to false. This allows the report to remain in the verification queue until it is reviewed and processed through the administrative module. The validated information is then stored in the device_reports table through the DeviceReport model. After successful creation, the backend returns a success response containing the newly created device-report information. The overall reporting process can be summarized as follows: User Selects Lost/Found Report → Enters Device Information → Uploads Supporting Proof → Submits Report → Server-side Validation → Unique Report Code Generation → Proof File Storage → Database Record Creation → Pending Verification The Device Reporting Module therefore provides a structured mechanism for collecting reliable device information and preparing each report for subsequent searching, matching, verification, and recovery operations within the TraceBack system. The overall workflow of the Device Reporting Module, including user input, device information submission, proof document upload, backend validation, database storage, report generation, and administrative review, is illustrated in Fig. 3.14.

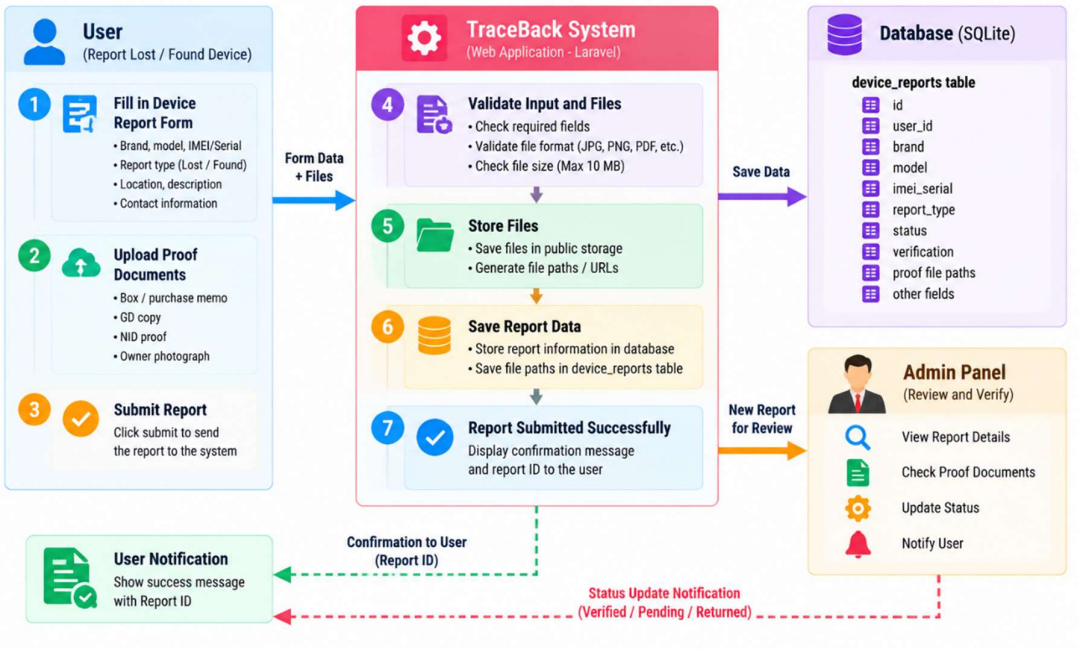


Fig. 3.14 Workflow of the Device Reporting Module

3.6 Device Search and Verification Module

The Device Search and Verification Module is a core component of the TraceBack Lost and Found System that allows stored device reports to be searched and checked using identifying information. The module helps users and administrators locate relevant lost or found device records and determine whether a matching report exists in the system. The search functionality retrieves records from the device_reports table. The system supports searching by device brand, model, IMEI or serial number, report code, and location. Search results can also be filtered according to the device category and report type, such as lost or found. The retrieved records are arranged according to their reporting time, with the most recently created records displayed first. A separate verification facility is provided for

checking a device using its IMEI/serial number or report code. When verification information is submitted, the request is processed by the Laravel backend through the verify function of the DeviceReportController. If no IMEI or serial value is provided, the system returns a validation response indicating that the required parameter is missing. When a valid value is submitted, the backend searches the device_reports records for matching IMEI/serial numbers or report codes. The system then returns the number of matching records together with the corresponding device information. This enables the frontend verification interface to inform the user whether relevant records are available in the TraceBack database. The frontend project also contains a dedicated verify page for this functionality. It is important to distinguish record matching from final administrative verification. A successful search only indicates that matching information exists in the database. The actual report status is managed separately and may be updated to pending, matched, verified, returned, or flagged. When an administrator changes the status to verified or returned, the report's verification field is also set to true. The overall process of the module can be summarized as: Enter Search or IMEI/Serial Information → Submit Request → Laravel Backend Processes Request → Search device_reports Database → Matching Records Identified → Results Returned to Frontend → Administrative Verification if Required. The device search and verification workflow is illustrated in Fig. 3.15.

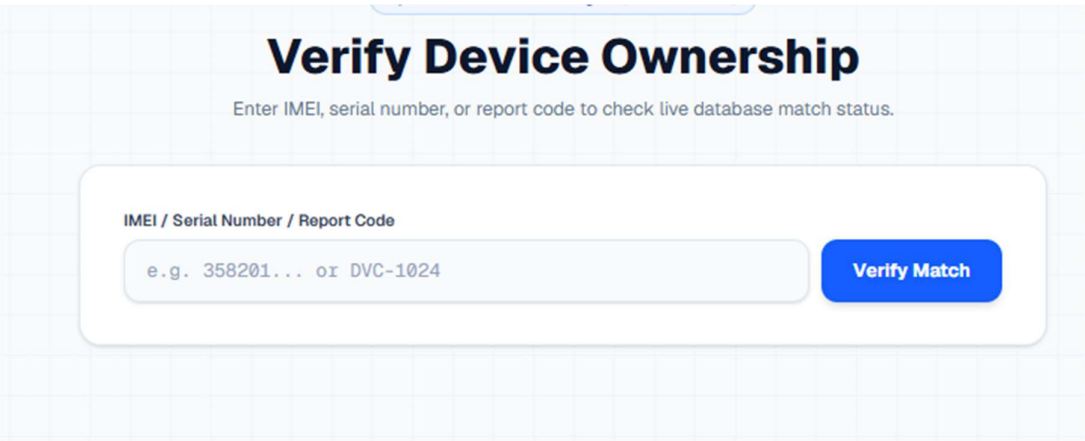


Fig. 3.15 Device Search and Verification Interface of the TraceBack System

3.7 Administrative Management

The Administrative Management Module provides a centralized platform for monitoring and controlling the major activities of the TraceBack Lost and Found System. Through the admin dashboard, authorized administrators can review device reports, monitor verification activities, manage users, and access system-related functions from a single interface. The dashboard presents an overall summary of system activities through several statistical cards, including Total Reports, Recovered Devices, Pending Claims, and Active Users. These indicators help administrators quickly understand the current condition of the system and identify records that require attention. A major component of the dashboard is the

Active Verification Queue. This section displays recently submitted device reports together with important information such as the report code, device name, IMEI or serial number, location, report type, current status, and available administrative actions. This allows administrators to review submitted records efficiently and proceed with verification or approval when necessary. The left-side navigation menu provides access to major administrative functions such as Dashboard, Device Reports, User Roster & Proofs, and System Settings. The Device Reports section enables administrators to manage lost and found reports, while the User Roster & Proofs section supports the review of user-related information and verification documents. The backend also allows administrators to update the status of a device report. Available report states include pending, matched, verified, returned, and flagged. When a report is marked as verified or returned, the corresponding verification value is also updated in the database. Administrators can also remove unnecessary or invalid reports when required. The administrative management and approval workflow is illustrated in Fig. 3.16.

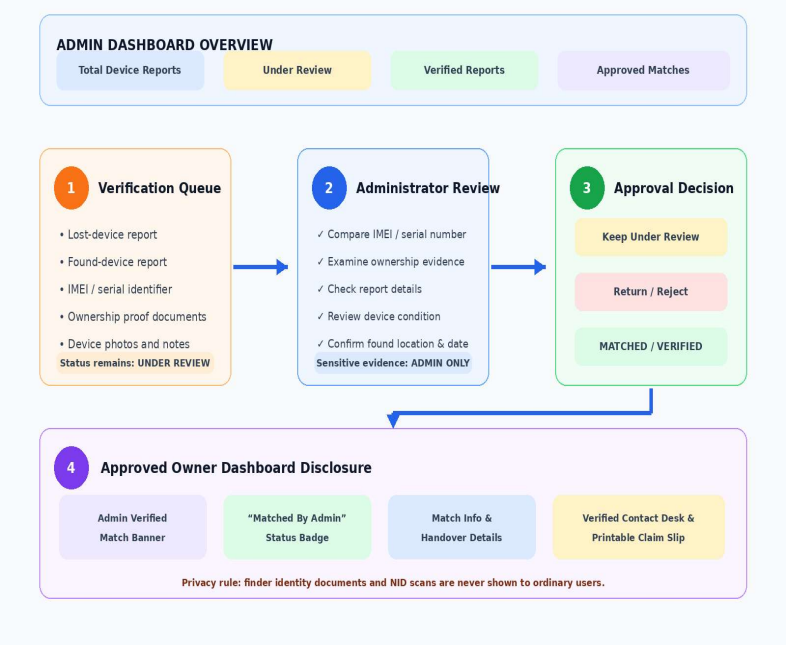


Fig. 3.16 Administrative Management and Approval Workflow of the TraceBack System

3.8 System Workflow

The TraceBack Lost and Found System follows a structured workflow connecting the frontend, Laravel backend, SQLite database, proof-document storage, verification functions, and admin panel. Users can report lost or found devices by providing device details, IMEI or serial number, location, contact information, and supporting proof documents. After submission, the Laravel backend validates the data, generates a unique report code, records the reporting date, sets the initial status to pending, and stores proof-file paths or URLs. The validated report is then saved in the device_reports table. Users can search or verify devices using information such as brand, model, IMEI/serial number,

report code, or location. Matching records are returned for review, while final verification is handled by the administrator. Through the TraceBack Admin Dashboard, administrators can review reports, monitor pending claims, and update report status to pending, matched, verified, returned, or flagged. When a report is marked as verified or returned, its verification state is updated accordingly. If the backend detects identical IMEI or serial-number values across lost and found reports, it creates a potential matching condition without publishing the matched information to the claimant. The administrator reviews both records and their supporting evidence. Until approval, the user-facing state remains Under Review. Once the administrator confirms the match and sets the status to matched or verified, the owner's dashboard receives the verified-match alert, comparison details, approved handover instructions, verified contact-desk information, and printable claim slip. This approval gate prevents an identifier match from being treated as automatic proof of ownership. The overall workflow of the TraceBack Lost and Found System, including user activities, backend processing, database operations, verification functions, and administrative actions, is illustrated in Fig. 3.17.

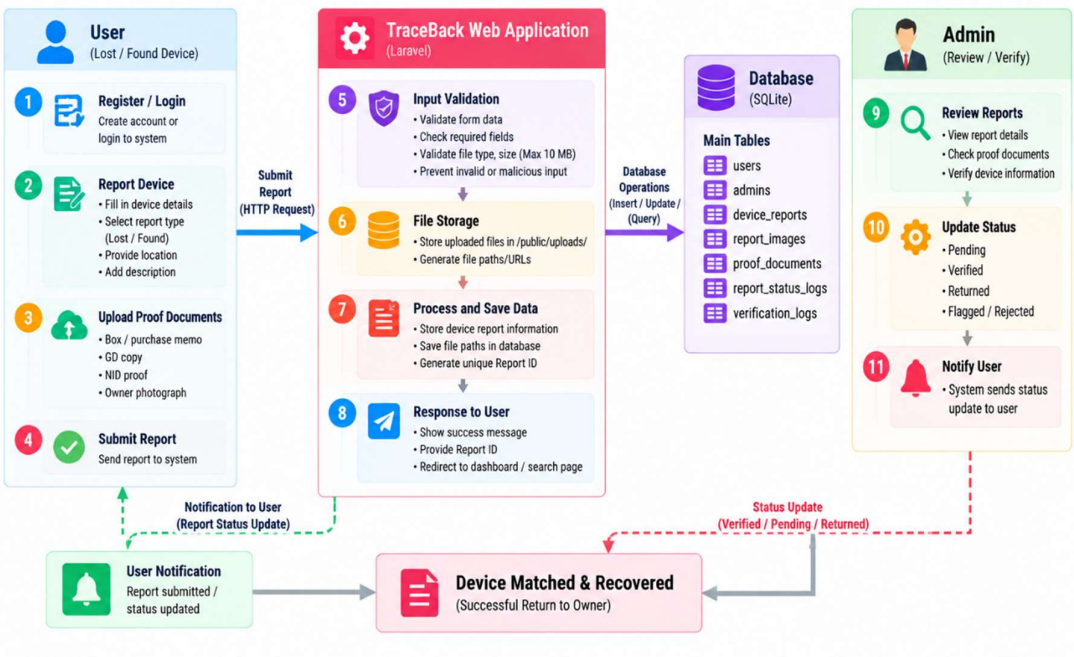


Fig. 3.17 Overall System Workflow of the TraceBack Lost and Found System

CHAPTER 4
IMPLEMENTATION AND CONFIGURATION

4.1 Development Environment

The TraceBack Lost and Found System was developed using a modern web development environment consisting of separate frontend and backend technologies. The frontend was developed using Next.js and JavaScript, while Tailwind CSS was used for interface styling. The backend was implemented using PHP and Laravel 12, which provides routing, request processing, validation, database interaction, file handling, and administrative functionality. The system uses SQLite as its primary database. Laravel Eloquent ORM is used for interaction between the application models and database records. The administrative interface is implemented using Laravel Blade templates, while Laravel's file-storage mechanism is used to manage uploaded verification documents. The development environment and technology stack are summarized in Table 4.1.

Technology / Component	**Role in the System**
Next.js	Used to develop the frontend web application
JavaScript	Provides frontend functionality and user interaction
Tailwind CSS	Used for styling and responsive interface design
Node.js / npm Environment	Supports frontend development, dependencies, and build processing
PHP	Server-side programming language used by the backend
Laravel 12	Backend framework for routing, validation, controllers, models, APIs, and administration
SQLite	Primary relational database used to store system records
Laravel Eloquent ORM	Provides model-based interaction with the database
Laravel Blade	Used to implement the administrative web interface
Composer	Manages PHP and Laravel dependencies
Laravel File Storage	Stores uploaded proof and verification documents.

Table 4.1 Development Environment and Technology Stack

4.2 Frontend Implementation

The frontend of the TraceBack Lost and Found System is implemented using Next.js with an App Router-based project structure. The frontend provides the user-facing interfaces required for accessing the system, submitting lost and found device reports, browsing device records, and verifying devices using identification information. The frontend source code is organized mainly under the src/app directory. Individual application routes are implemented using separate pages. TSX files help keep the major system functions organized and maintainable. The project also contains reusable components for common

interface elements such as the navigation bar, hero section, system workflow information, recent device information, and footer.

The authenticated /dashboard route implements the owner-facing recovery interface. Reports awaiting administrative decision are labeled Under Review. For an administrator-approved matched or verified report, the page displays an Admin Verified Match banner, a "Matched By Admin" badge, a Match Info button with a side-by-side handover-details modal, verified contact-desk information, and a Print Claim Slip action. The interface deliberately excludes the finder's NID scan and other sensitive identity documents. The main frontend pages are summarized in Table 4.2.

Route	Source Page	Main Purpose
/	src/app/page.tsx	Main landing page of the TraceBack system
/devices	src/app/devices/page.tsx	Displays and searches device records
/lost	src/app/lost/page.tsx	Provides the lost-device reporting interface
/found	src/app/found/page.tsx	Provides the found-device reporting interface
/verify	src/app/verify/page.tsx	Provides device verification functionality
/admin/dashboard	src/app/admin/dashboard/page.tsx	Frontend admin dashboard interface
/admin/users	src/app/admin/users/page.tsx	Displays user-related administrative information.
/dashboard	src/app/dashboard/page.tsx	Displays user reports, approved match alerts, handover details, and claim-slip printing

Table 4.2 Main Frontend Pages of the TraceBack System

4.3 User Authentication Implementation

The TraceBack system implements user authentication to control access to protected functions such as lost-device reporting, found-device reporting, and the user dashboard. New users can create an account through the registration interface, while existing users can log in using their registered credentials. After successful authentication, the user can access the protected reporting forms and manage submitted device reports. If an unauthenticated user attempts to open a protected reporting page, the system displays a Citizen Account Required warning instead of showing the submission form. The user is then provided with options to sign in or create a new account. This mechanism helps associate device reports with authenticated users and reduces unauthorized or fraudulent submissions. The account-required warning displayed on the lost-device reporting page is shown in Fig. 4.1.

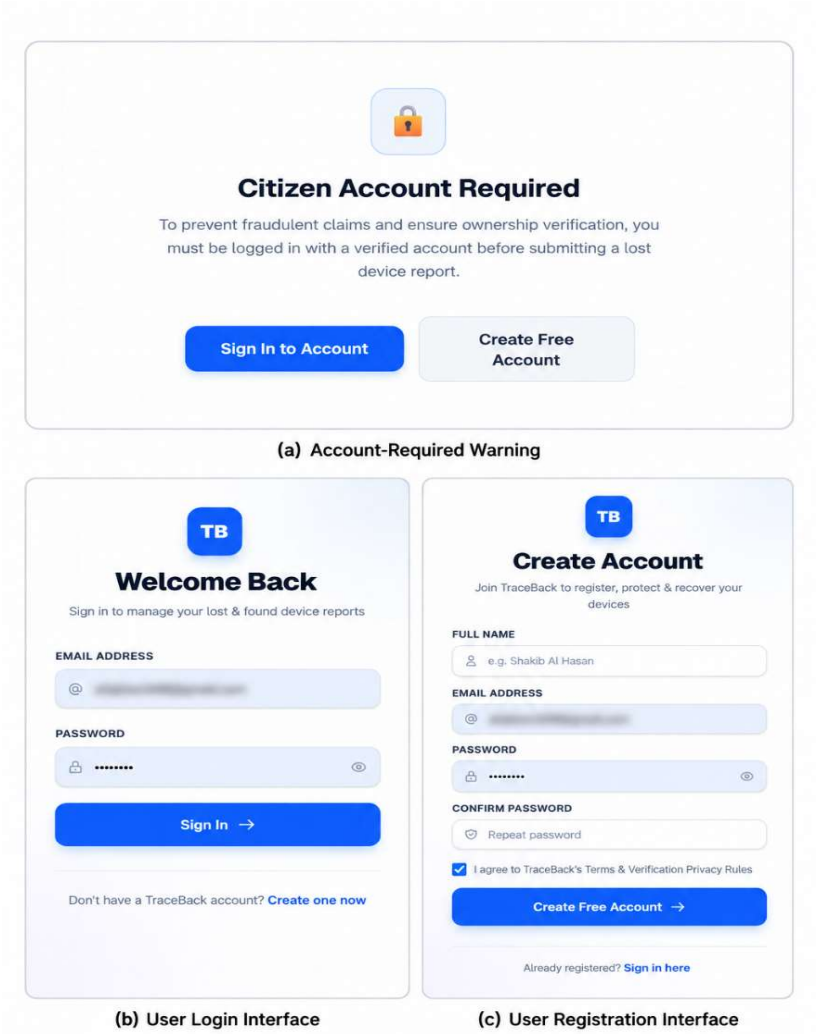


Fig. 4.1 User Authentication Interfaces: (a) Account-Required Warning, (b) User Login Interface, and (c) User Registration Interface

4.4 Backend Implementation

The backend of the TraceBack Lost and Found System is implemented using Laravel 12 and PHP. It is responsible for processing requests received from the frontend, validating submitted data, communicating with the database, handling proof-document uploads, managing device-report status, and supporting administrative operations. The backend follows Laravel's structured application architecture. Controllers are used to process incoming requests, models are used to interact with database records, routes connect HTTP requests with appropriate backend functions, and Blade templates are used to implement the administrative interface. The backend also enforces the administrator-approval gate for device matching. A lost/found IMEI or serial-number match is processed as a potential match, while user-facing disclosure is conditioned on an administrator-approved matched or verified status. The dashboard response exposes approved match and handover

information only to the authenticated owner associated with the lost-device report. The main backend components are summarized in Table 4.3.

Backend Component	Implementation Role
DeviceReportController	Handles report listing, submission, verification, status update, and deletion
DeviceReport Model	Represents lost and found device records in the database
User Model	Manages user-related account information
API Routes	Connects frontend requests with backend controller methods.
Laravel Validation	Validates device information and uploaded proof files
Eloquent ORM	Provides model-based database interaction
SQLite	Stores device reports, users, sessions, and supporting records
Laravel File Storage	Stores uploaded proof documents.
Blade Templates	Implements the administrative dashboard and management pages

Table 4.3 Main Backend Components of the TraceBack System

4.5 Database Implementation

The TraceBack Lost and Found System database is implemented using SQLite and managed through Laravel's database configuration, migrations, and Eloquent ORM. SQLite is configured as the default database connection, with the database file stored within the Laravel application's database directory. Laravel migrations are used to create and maintain the required database tables. The main application table is `device_reports`, which stores information about lost and found devices. The implemented fields include the report code, brand, model, category, report type, IMEI or serial number, color, location, reporting date, status, description, contact information, verification state, and timestamps. The `code` field is unique, while the `IMEI` field is indexed to support efficient searching. A separate migration extends the `device_reports` table by adding four proof-document fields: `box_memo_proof`, `gd_copy_proof`, `nid_proof`, and `owner_pic_proof`. These fields store the paths or URLs of uploaded verification documents rather than the actual binary files. The database also contains Laravel-generated tables such as `users`, `password_reset_tokens`, and `sessions` for account and authentication-related functions. Additional tables including `cache`, `cache_locks`, `jobs`, `job_batches`, and `failed_jobs` support internal Laravel operations. The implemented technical database architecture is illustrated in Fig. 4.2. The main database migrations are summarized in Table 4.4.

Migration	Purpose
create_users_table	Creates user, password-reset, and session-related tables
create_cache_table	Creates cache and cache-lock tables
create_jobs_table	Creates queue and failed-job tables
create_device_reports_table	Creates the main lost and found device-report table
add_proof_documents_to_device_reports_table	Adds proof-document fields to device reports

Table 4.4 Main Database Migrations

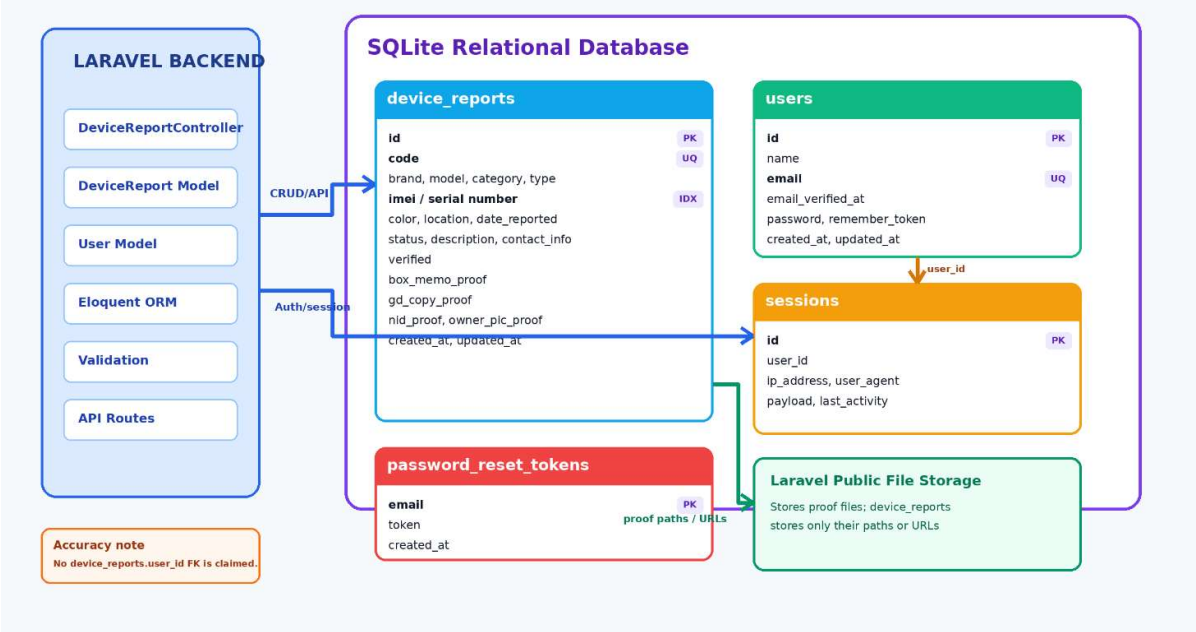


Fig. 4.2 Technical Database Schema Architecture of the TraceBack System

4.6 Device Report Model

The DeviceReport model represents the device_reports table in the TraceBack Lost and Found System. It is implemented using Laravel's Eloquent ORM and acts as the connection between the backend application logic and the device-report records stored in the database. The model uses Laravel's HasFactory trait and defines the fields that are allowed for mass assignment through the $fillable property. These fields include device information, report status, contact details, verification state, and proof-document references. The implementation of the DeviceReport model, which represents the device_reports table and defines the database fields used by the system, is shown in Code Snippet 4.1.

```php
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class DeviceReport extends Model
{
    use HasFactory;
    protected $fillable = [
        'code',
        'brand',
        'model',
        'category',
        'type',
        'imei',
        'color',
        'location',
        'date_reported',
        'status',
        'description',
        'contact_info',
        'verified',
        'box_memo_proof',
        'gd_copy_proof',
        'nid_proof',
        'owner_pic_proof',
    ];
}
```

Code Snippet 4.1 DeviceReport Model

4.7 Device Report Controller

The DeviceReportController is the main backend controller responsible for managing device-report operations in the TraceBack Lost and Found System. It receives requests from the frontend or administrative interface, processes the submitted information, communicates with the DeviceReport model, and returns the required response. The controller contains five major methods: index(), store(), verify(), updateStatus(), and destroy(). The index() method retrieves device reports from the database. It supports filtering by category and report type (lost or found). It also provides search functionality using device brand, model, IMEI/serial number, report code, or location. The retrieved records are ordered by creation time and returned using pagination. The store() method handles the submission of a new device report. It validates required information such as brand, model, category, type, IMEI/serial number, and location. It also validates uploaded proof documents. After successful validation, the controller generates a unique report code, records the reporting date, sets the initial status to pending, and sets the verification value to false. The validated data is then stored in the database through the DeviceReport model. The verify() method is used to search for matching device records. It accepts an IMEI/serial number or report code and searches the device_reports table for matching records. The response contains the number of matches and the corresponding device information. The updateStatus() method allows the status of an existing device report to be changed. Supported status values include pending, matched, verified, returned, and flagged. When a report is marked as verified or returned, the controller also sets its verified field to true. The destroy() method removes a selected device report from the database. This function can be used by the administrative interface when an invalid or unnecessary record needs to be deleted. For the approval-based recovery workflow, the controller and related dashboard logic distinguish database matching from administrative confirmation. Status changes control whether the verified-match banner, "Matched By Admin" badge, Match Info modal, handover instructions, contact-desk information, and printable claim slip become available to the owner. Records that have not been approved continue to return an Under Review state. The main controller methods are summarized in Table 4.5.

Method	**Main Function**
index()	Retrieves, filters, searches, and paginates device reports
store()	Validates and stores a new lost/found report
verify()	Searches records using IMEI/serial number or report code
updateStatus()	Updates report status and verification state
destroy()	Deletes a selected device report
updateStatus() / dashboard logic	Controls admin approval and owner-only disclosure of verified match information

Table 4.5 Main Methods of DeviceReportController

The main processing steps performed by the DeviceReportController—from receiving and validating a report to storing it, searching for matches, and updating its administrative status—are presented in Algorithm 4.1.

Start

Step 1: Receive device report request

Step 2: Validate required device information

Step 3: Validate uploaded proof documents

Step 4: If validation fails, return an error

Step 5: Generate a unique device report code

Step 6: Set status = pending

Step 7: Set verified = false

Step 8: Store proof files and save their URLs

Step 9: Save the report in the device_reports table

Step 10: Allow searching by IMEI/serial number or report code

Step 11: Return matching records if found

Step 12: Administrator reviews the report

Step 13: Update status when required

Step 14: If status is verified or returned, set verified = true

End

Algorithm 4.1 Device Report Processing

4.8 API Routes

The API routes of the TraceBack Lost and Found System provide the communication layer between the frontend application and the Laravel backend. These routes are defined in the `routes/api.php` file and connect incoming HTTP requests with the appropriate methods of the `DeviceReportController`. The implemented API routes support device listing and searching, report submission, IMEI or serial-number verification, report-status updates, and backend health checking. The implemented API routes are summarized in Table 4.6.

HTTP Method	Route	Action	Purpose
GET	/health-check	Closure	Checks whether the Laravel API and SQLite database connection are active
GET	/devices	index()	Retrieves device reports with search, filtering, and pagination
POST	/reports	store()	Submits and stores a new lost or found device report

HTTP Method	Route	Action	Purpose
POST	/verify-imei	verify()	Searches for a device using IMEI/serial number or report code
PATCH	/devices/{id}/status	updateStatus()	Updates the status and verification state of a selected report

Table 4.6 Main API Routes of the TraceBack System

4.9 Admin Panel

The Admin Panel of the TraceBack Lost and Found System is implemented using Laravel Blade templates. It provides a centralized administrative interface for monitoring device reports, reviewing verification activities, managing report status, and accessing user-related information. The main dashboard displays summary information such as Total Reports, Recovered Devices, Pending Claims, and Active Users. These values are retrieved from the backend and displayed dynamically through Blade variables. The dashboard also contains an Active Verification Queue, where recent device reports are presented with their report code, device information, IMEI or serial number, location, report type, and current status. The administrative layout contains navigation options for Dashboard, Device Reports, User Roster & Proofs, and System Settings. A common Blade layout is used to maintain a consistent sidebar, header, styling, and navigation structure across the administrative pages. Administrators can manage device-report status directly from the administrative interface. The backend accepts the status values `pending`, `matched`, `verified`, `returned`, and `flagged`. When a report is changed to `verified` or `returned`, the system also changes its `verified` value to `true`. Reports that are no longer required can also be removed through the backend delete operation. The administrator can compare a candidate lost report with its corresponding found report, review the IMEI or serial number and proof documents, and approve or reject the potential match. Approval changes the controlled status to matched or verified and enables the owner-facing recovery information. Finder NID scans and other sensitive proof files remain visible only within the protected administrative interface. The main administrative functions are summarized in Table 4.7.

Admin Function	Implementation Purpose
Dashboard Overview	Displays overall report, recovery, pending, and user statistics
Verification Queue	Displays recently submitted reports for administrative review.
Device Reports	Provides access to lost and found device records
Status Management	Updates reports as pending, matched, verified, returned, or flagged
Report Approval	Allows eligible reports to be marked as verified
User Roster & Proofs	Provides access to user and proof-related administrative information
Report Deletion	Removes unnecessary device-report records
System Settings	Provides access to administrative system configuration
Match Approval & Handover	Reviews candidate pairs and activates owner-facing match and claim information

Table 4.7 Main Functions of the Admin Panel

4.10 File Upload and Proof Handling

The File Upload and Proof Handling mechanism of the TraceBack Lost and Found System is implemented in the Laravel backend to manage supporting documents submitted with device reports. These documents assist the administrative verification process by providing evidence related to device ownership and reporting. The system supports four proof fields: box or purchase memo, GD copy, NID proof, and owner photograph. Laravel performs server-side validation before accepting the uploaded files. Each proof document is optional and may be submitted in several supported image or document formats. The maximum permitted size for each uploaded file is 10 MB. The proof fields and validation rules are summarized in Table 4.8.

Proof Field	Purpose	Accepted Formats	Maximum Size
box_memo_proof	Box memo or purchase receipt	JPG, JPEG, PNG, WEBP, GIF, BMP, PDF, SVG	10 MB
gd_copy_proof	Police General Diary (GD) copy	JPG, JPEG, PNG, WEBP, GIF, BMP, PDF, SVG	10 MB
nid_proof	National ID document	JPG, JPEG, PNG, WEBP, GIF, BMP, PDF, SVG	10 MB
owner_pic_proof	Photograph of the device owner	JPG, JPEG, PNG, WEBP, GIF, BMP, PDF, SVG	10 MB

Table 4.8 Proof Document Fields and Validation

CHAPTER 5
TESTING AND VALIDATION

5.1 Testing Strategy

The TraceBack Lost and Found System was evaluated through a functional testing strategy focusing on the major operations implemented in the frontend and Laravel backend [10]. The purpose of testing was to verify that each system function produced the expected response for both valid and invalid inputs and that information was correctly processed, stored, retrieved, and updated. The testing process covered major functions including API health checking, device listing and searching, lost and found report submission, IMEI/serial verification, report-status updates, administrative report management, and proof-file validation. These functions correspond to the primary operations implemented within the TraceBack backend. Both positive and negative test scenarios were considered [10]. Positive testing was used to confirm that valid inputs produced the expected results, while negative testing was used to check the system's response to missing, invalid, or unsupported input. The expected result of each test was compared with the observed system response to determine whether the functionality operated correctly [31]. The testing approaches are summarized in Table 5.1.

Testing Approach	**Purpose**
Functional Testing	Verifies that individual system functions operate correctly
API Testing	Checks communication between the frontend and Laravel backend
Input Validation Testing	Tests valid, missing, and invalid user inputs
Database Testing	Verifies storage, retrieval, and updating of device records
File Validation Testing	Checks proof-file type, size, and storage handling
Administrative Testing	Verifies report review, status updates, and management operations

Table 5.1 Testing Strategy of the TraceBack System

5.2 Authentication Testing

Authentication testing was conducted to verify the registration, login, protected-page access, and logout functions of the TraceBack system. The tests examined whether valid users could create accounts and log in, while invalid or incomplete information produced appropriate validation messages. Protected reporting pages were also tested without authentication. In this case, the system correctly displayed the Citizen Account Required warning and provided options to sign in or create an account. The authentication test results are presented in Table 5.2.

Test Case	Test Action	Expected Result	Status
Valid Registration	Submit valid user information	Account is created successfully	Passed
Empty Required Field	Leave a required field empty	Validation message is displayed	Passed
Password Mismatch	Enter different passwords	Registration is rejected	Passed
Duplicate Email	Register with an existing email	Duplicate-email warning is displayed	Passed
Valid Login	Enter correct email and password	User dashboard is opened	Passed
Invalid Login	Enter incorrect credentials	Login error is displayed	Passed
Protected Page Access	Open reporting page without login	Account-required warning is displayed	Passed
Logout	Select the logout option	User session ends successfully	Passed

Table 5.2 Authentication Test Results

5.3 Health Check Testing

Health check testing was performed to verify that the Laravel backend API of the TraceBack system was available and capable of responding to requests from the frontend. The backend provides a dedicated GET /api/health-check endpoint for this purpose. When the endpoint is accessed successfully, the server returns a JSON response containing a success status, a confirmation message, and the current timestamp. The frontend can therefore use this endpoint to determine whether communication with the Laravel backend is active. The health-check test cases are summarized in Table 5.3.

Test Case	Request	Expected Result	Validation
Backend Availability	GET /api/health-check	Server returns a successful JSON response	API is reachable
Response Status	Health-check request	Status returns success	Response is valid
Response Message	Health-check request	Backend confirmation message is returned	API is operational
Timestamp	Health-check request	Current timestamp is included	Server generates response correctly

Table 5.3 Health Check Test

5.4 Device Listing and Search Testing

The Device Listing and Search Testing was performed to verify that stored device reports could be retrieved, searched, filtered, and displayed correctly. The Laravel backend processes these operations through the index() method of the DeviceReportController. The system supports searching device records using brand, model, IMEI/serial number, report code, and location. It also allows filtering according to device category and report type such as lost or found. The retrieved records are ordered by the most recently created reports and returned using pagination. The device listing and search test cases are summarized in Table 5.4.

Test Case	Test Input / Action	Expected Result	Status
Display Device List	Open device listing page	Stored device records are displayed	Passed
Brand Search	Enter a valid device brand	Matching brand records are displayed	Passed
Model Search	Enter an existing device model	Matching model records are displayed	Passed
IMEI/Serial Search	Enter a stored IMEI or serial number	Corresponding device record is displayed	Passed
Report Code Search	Enter a valid report code	Matching report is displayed	Passed
Location Search	Enter a reported location	Relevant records are displayed	Passed
Category Filter	Select a device category	Only records from the selected category are displayed	Passed
Report Type Filter	Select lost or found	Only the selected report type is displayed	Passed
Invalid Search	Enter a value with no matching record	No matching records are returned	Passed

Table 5.4 Device Listing and Search Test Cases

5.5 Report Submission Testing

The Report Submission Testing was performed to verify that users could successfully submit lost or found device reports and that the Laravel backend correctly validated and stored the submitted information. The report submission process is handled by the store() method of the DeviceReportController. The backend requires important fields such as brand, model, category, report type, IMEI/serial number, and location. The type field accepts only lost or found. Optional fields include color, description, contact information, and supporting proof documents. After successful validation, the system generates a report code, records the reporting date, assigns the initial status as pending, sets verified to false, and stores the report in the device_reports table. The report submission test cases are summarized in Table 5.5.

Test Case	Test Input / Action	Expected Result	Status
Valid Lost Report	Submit all required fields with type = lost	Report is successfully stored	Passed
Valid Found Report	Submit all required fields with type = found	Report is successfully stored	Passed
Missing Brand	Submit report without brand	Validation error is returned	Passed
Missing Model	Submit report without model	Validation error is returned	Passed
Missing IMEI/Serial	Submit report without IMEI/serial number	Validation error is returned	Passed
Missing Location	Submit report without location	Validation error is returned	Passed
Invalid Report Type	Submit a value other than lost or found	Validation error is returned	Passed
Report Code Generation	Submit a valid report	Unique DVC-XXXX code is generated	Passed
Initial Status	Submit a valid report	Status is stored as pending	Passed
Verification State	Submit a valid report	Verified is stored as false	Passed
Database Storage	Complete valid submission	A new record is created in device_reports	Passed

Table 5.5 Report Submission Test Cases

5.6 IMEI/Serial Verification Testing

The IMEI/Serial Verification Testing was designed to verify whether the TraceBack system can correctly identify stored device records using an IMEI/serial number or report code. The verification process is handled by the verify() method of the DeviceReportController. When a verification value is submitted, the backend searches the device_reports table for matching records in either the imei or code field. If the input is empty, the system returns a validation response with HTTP status 422. When matching records are found, the API returns the number of matches together with the corresponding device information. Additional workflow testing confirmed that an identifier match alone does not expose found-device or handover information on the user's dashboard. Before administrative approval, the report remains Under Review. After the administrator sets an eligible record to matched or verified, the correct owner can view the verified-match alert, matching-device comparison, claim instructions, helpdesk information, and print option. The IMEI/Serial verification test cases are summarized in Table 5.6.

Test Case	Test Input / Action	Expected Result
Valid IMEI	Enter an existing IMEI number	Matching device record is returned
Valid Serial Number	Enter an existing stored serial number	Corresponding device record is returned

Test Case	Test Input / Action	Expected Result
Valid Report Code	Enter an existing DVC-XXXX code	Matching report is returned
Non-existing IMEI	Enter an IMEI not stored in the database	No matching records are returned
Invalid Report Code	Enter a non-existing report code	No matching records are returned
Empty Input	Submit without IMEI/serial or code	Validation message is returned with status 422
Multiple Matches	Enter a value matching multiple records	Matching records and total count are returned

Table 5.6 IMEI/Serial Verification Test Cases

5.7 Status Update Testing

The Status Update Testing evaluates whether the TraceBack system correctly updates the processing status of an existing device report. This operation is handled by the updateStatus() method of the DeviceReportController. The implemented system supports five status values: pending, matched, verified, returned, and flagged. When a valid status is submitted, the selected device_reports record is updated and saved in the database. In addition, when the status is changed to either verified or returned, the system automatically sets the verified field to true. The status update test cases are summarized in Table 5.7.

Test Case	Test Input / Action	Expected Result
Set Pending	Update report status to pending	Report status becomes pending
Set Matched	Update report status to matched	Report status becomes matched
Set Verified	Update report status to verified	Status becomes verified and verified = true
Set Returned	Update report status to returned	Status becomes returned and verified = true
Set Flagged	Update report status to flagged	Report status becomes flagged
Invalid Status	Submit a status outside the supported values	Existing report status remains unchanged
Invalid Report ID	Submit status update for a non-existing report	Report lookup fails, and the update is not performed
Unapproved Match	Create matching lost/found IMEI records without admin approval	Owner dashboard remains Under Review and hides handover information
Approved Match	Administrator sets an eligible match to matched or verified	Owner dashboard displays verified match and claim information

Table 5.7 Status Update Test Cases

5.8 Admin Report Management Testing

The Admin Report Management Testing was designed to verify that the administrative functions of the TraceBack Lost and Found System operate correctly. Through the admin panel, administrators can review device reports, search and filter records, update report status, approve verification requests, and remove unnecessary reports. These operations are supported by the Laravel backend and DeviceReportController. The testing focuses on ensuring that administrative actions correctly affect the corresponding records stored in the device_reports table. The tests also checked the approval boundary and privacy controls. Administrators could review a lost/found candidate pair and approve it, while an ordinary user could not view sensitive finder identity documents. The owner-facing dashboard displayed match and handover information only after the approved status was saved. The admin report management test cases are summarized in Table 5.8.

Test Case	Test Input / Action	Expected Result
View Reports	Open the Device Reports page	Available device reports are displayed
Search Report	Search by brand, model, IMEI, code, or location	Relevant matching records are displayed
Filter Lost Reports	Select the lost report type	Only lost-device reports are displayed
Filter Found Reports	Select the found report type	Only found-device reports are displayed
Approve Report	Mark an eligible report as verified	Status changes to verified and verification state becomes true
Update Status	Select a valid report status	Selected report is updated successfully
Delete Report	Delete an existing report	Selected record is removed from the database
Invalid Report ID	Perform an action on a non-existing report	Record lookup fails, and no incorrect record is modified
Review Candidate Pair	Open matching lost and found records	Side-by-side report and proof information is available to the administrator
Protect Finder Identity	Open approved match as an ordinary user	Finder NID scan and sensitive proof files are not displayed

Table 5.8 Admin Report Management Test Cases

5.9 File Upload Validation Testing

The File Upload Validation Testing was designed to verify that the TraceBack system correctly validates supporting proof documents submitted with device reports. The backend supports four proof fields: box_memo_proof, gd_copy_proof, nid_proof, and owner_pic_proof. Each proof document is optional, but when a file is uploaded, Laravel validates its file type and size before storing it. The accepted formats are JPG, JPEG, PNG,

WEBP, GIF, BMP, PDF, and SVG, with a maximum size of 10 MB per file. Valid files are stored in the public proofs directory, and the generated file URL is saved with the corresponding device report. The file upload validation test cases are summarized in Table 5.9.

Test Case	Test Input / Action	Expected Result
Valid JPG Upload	Upload a JPG proof file within 10 MB	File is accepted and stored successfully
Valid PNG Upload	Upload a PNG proof file within 10 MB	File is accepted and stored successfully
Unsupported Format	Upload an unsupported file format	A validation error is returned
Oversized File	Upload a proof file larger than 10 MB	A validation error is returned
No Proof File	Submit report without a proof file	Report can still be processed because proof fields are optional
Multiple Proof Files	Upload valid files for several proof fields	Each valid file is processed and stored
File URL Storage	Upload a valid proof document	The generated storage URL is saved in the report record

Table 5.9 File Upload Validation Test Cases

5.10 Test Summary and Results

The testing and validation activities of the TraceBack Lost and Found System covered the major functions implemented in the frontend and Laravel backend. The test cases were designed to evaluate backend availability, device listing and searching, report submission, IMEI/Serial verification, status management, administrative operations, and proof-file validation. The test results demonstrate that the implemented system functions operated according to the defined requirements. The backend successfully responded to health-check requests, device records were retrieved and searched correctly, lost and found reports were submitted and stored successfully, and IMEI/Serial Number verification returned the expected matching results. The status management, administrative report handling, and file upload validation features were also tested successfully. Valid files were accepted according to the defined validation rules, while invalid inputs and unsupported files generated appropriate validation responses. Overall, the testing results confirm that the major functional components of the TraceBack system operate correctly and satisfy the core requirements of device reporting, searching, verification, management, and proof-document handling.The approval-based matching tests passed: potential matches remained Under Review before approval; verified dashboard alerts and claim information appeared after approval; the printable claim slip was available to the rightful owner; and finder identity-document scans remained restricted from ordinary-user access. A summary of the system testing is presented in Table 5.10.

Test Area	Validation Objective	Expected Result	Execution Status
Health Check	Verify backend API availability	API returns a successful response	Passed
Device Listing & Search	Verify retrieval, search, filtering, and pagination	Relevant device records are returned	Passed
Report Submission	Verify validation and database storage	Valid report is stored successfully	Passed
IMEI/Serial Verification	Verify matching by IMEI, serial, or report code	Matching records are returned correctly	Passed
Status Update	Verify report-status modification	Valid status is updated correctly	Passed
Admin Report Management	Verify administrative report operations	Requested management action is completed	Passed
Approval-Based Matching	Verify approval gate, dashboard disclosure, and privacy controls	Information appears only after approval; sensitive finder files remain restricted	Passed

Table 5.10 Summary of System Testing

CHAPTER 6
RESULTS AND DISCUSSION

6.1 Functional Results

The TraceBack Lost and Found System successfully implements the major functions required for reporting, storing, searching, and managing lost and found electronic devices. Users can submit separate lost and found device reports by providing device information such as brand, model, category, IMEI or serial number, color, location, description, and contact details. After a valid report is submitted, the Laravel backend generates a unique report code in the format DVC-XXXX, records the reporting date, assigns the initial status as pending, and stores the report in the device_reports table. The system also supports proof-document references associated with submitted device reports. The implemented search functionality allows stored records to be searched using device brand, model, IMEI/serial number, report code, and location. Reports can also be filtered according to device category and report type (lost or found). In addition, the system maintains different report states, including pending, matched, verified, returned, and flagged. The functional results are summarized in Table 6.1.

Functional Area	**Result**
Lost Device Reporting	Successfully implemented
Found Device Reporting	Successfully implemented
Report Code Generation	Unique DVC-XXXX code generated
Database Storage	Device reports stored in the database
Device Search	Records searchable using multiple attributes.
Category Filtering	Reports can be filtered by category.
Lost/Found Filtering	Reports can be filtered by report type.
Status Management	Multiple report statuses are supported.
Proof Handling	Proof-document references are stored with reports.
Admin Verified Match Alert	Displayed to the rightful owner after administrative approval
Handover Details & Claim Slip	Approved details are shown, and the claim slip can be printed.

Table 6.1 Functional Results of the TraceBack System

6.1.1 User Registration and Login Result

The implemented registration and login functions allow users to create an account and access the protected features of the TraceBack system. During registration, the system validates the user's name, email address, password, and password confirmation before creating the account. Registered users can then log in using valid credentials and access

their dashboard, lost-device reporting, and found-device reporting facilities. If an unauthenticated user attempts to access a protected reporting page, the system displays a Citizen Account Required warning and provides options to sign in or create an account. Invalid login credentials and incomplete registration information are rejected with appropriate validation messages. These results demonstrate that the authentication system successfully restricts protected functions to registered users and supports secure association between users and submitted device reports. The successful implementation of the registration, login, and protected-access functions is illustrated in Fig. 6.1.

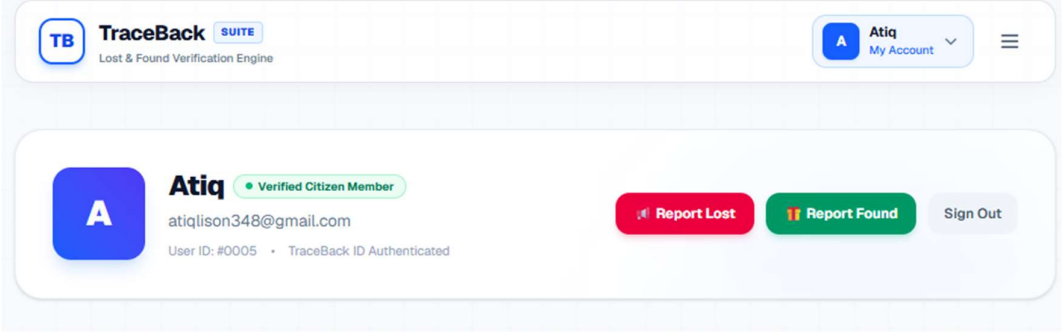


Fig. 6.1 User Registration, Login, and Protected-Access Results

6.1.2 Lost Device Reporting Result

The implemented Lost Device Reporting function allows authenticated users to submit information about a lost device through the TraceBack system. Users can provide important device details, including device category, brand, model, IMEI or Serial Number, color, last known location, contact information, and description. The interface also supports the uploading of ownership-related proof documents, such as Box/Memo Proof, GD Copy Proof, NID Proof, and Owner Picture Proof. After the required information is submitted, the Laravel backend validates the entered data and uploaded files. If the submission is valid, the system generates a unique report code in the format DVC-1119, records the reporting date, assigns the initial status as pending, and stores the report in the database. Invalid or incomplete submissions are rejected with appropriate validation messages. These results demonstrate that the Lost Device Reporting function successfully collects, validates, and stores lost-device information for subsequent searching, verification, and administrative review. The successful lost-device reporting result is illustrated in Fig. 6.2.

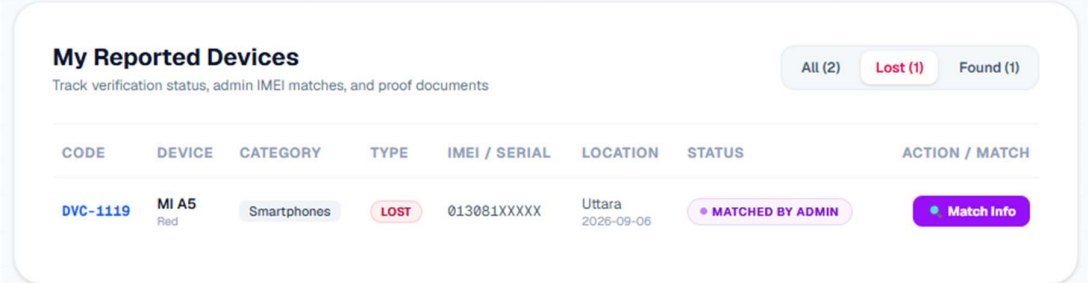


Fig. 6.2 Lost Device Reporting Result

6.1.3 Lost Device Search Result

The implemented Lost Device Search function allows users to search previously reported lost devices stored in the TraceBack system. Users can search device records using information such as brand, model, IMEI or Serial Number, report code, and location. The Laravel backend processes the submitted search request and retrieves matching records from the device_reports table. Relevant lost-device information, including device category, brand, model, masked IMEI or Serial Number, location, report status, and available verification information, is then displayed to the user. If no matching record is found, the system returns an appropriate no-result response. The search function can also be combined with report-type filtering so that only lost-device records are displayed. These results demonstrate that the TraceBack system can successfully retrieve and display lost-device reports using multiple search attributes. The successful lost-device search result is illustrated in Fig. 6.3.

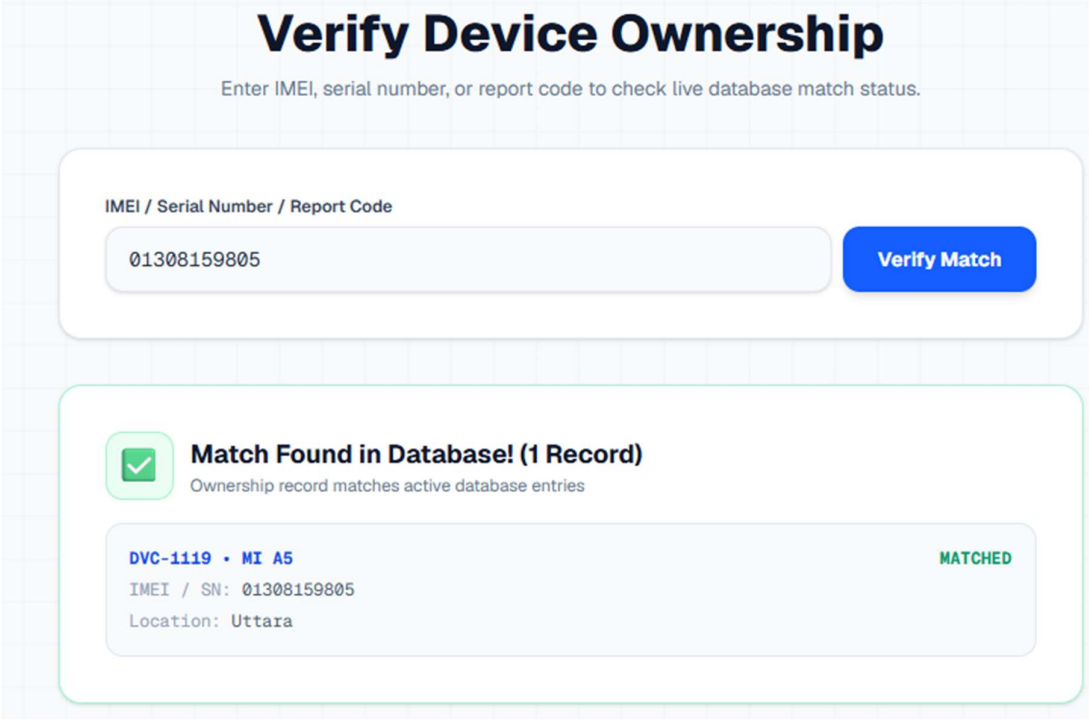


Fig. 6.3 Lost Device Search Result

6.1.4 Found Device Reporting Result

The implemented Found Device Reporting function allows authenticated users to submit information about a device that they have found through the TraceBack system. Users can provide important details such as device category, brand, model, IMEI or Serial Number, color, found location, contact information, description, and available device images. After submission, the Laravel backend validates the required information and uploaded files before storing the report in the device_reports table. For a valid submission, the system

generates a unique report code in the format DVC-1124, records the reporting date, assigns the initial status as pending, and stores the report for subsequent searching, matching, and administrative review. Invalid or incomplete submissions are rejected with appropriate validation messages. These results demonstrate that the Found Device Reporting function successfully collects, validates, and stores found-device information in an organized manner. The successful found-device reporting result is illustrated in Fig. 6.4.

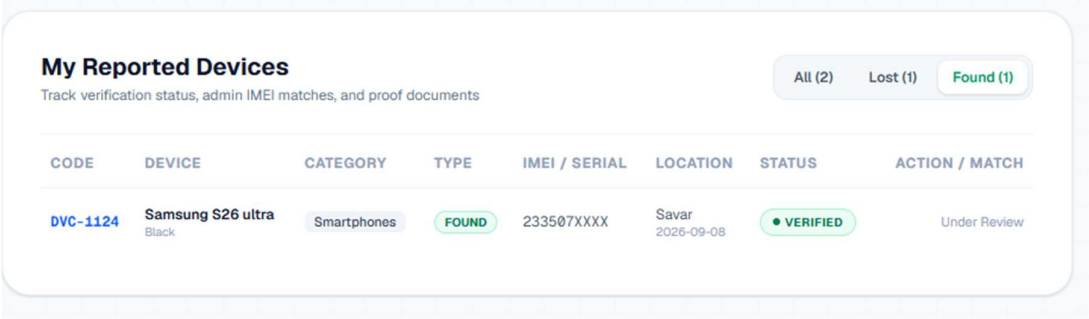


Fig. 6.4 Found Device Reporting Result

6.1.5 Submitted Found Device Reporting Result

After a found-device report is successfully submitted, the TraceBack system stores the submitted information in the device_reports table and makes the report available for subsequent review and management. The stored record contains the submitted device details, generated DVC-1124 report code, reporting date, location, report type, and current status. Newly submitted found-device reports are initially assigned the status pending and remain available for administrative review and future matching with corresponding lost-device reports. The submitted report can then be retrieved through the system's device listing and management functions. These results confirm that the TraceBack system successfully preserves and displays submitted found-device information after report creation. The submitted found-device reporting result is illustrated in Fig. 6.5.

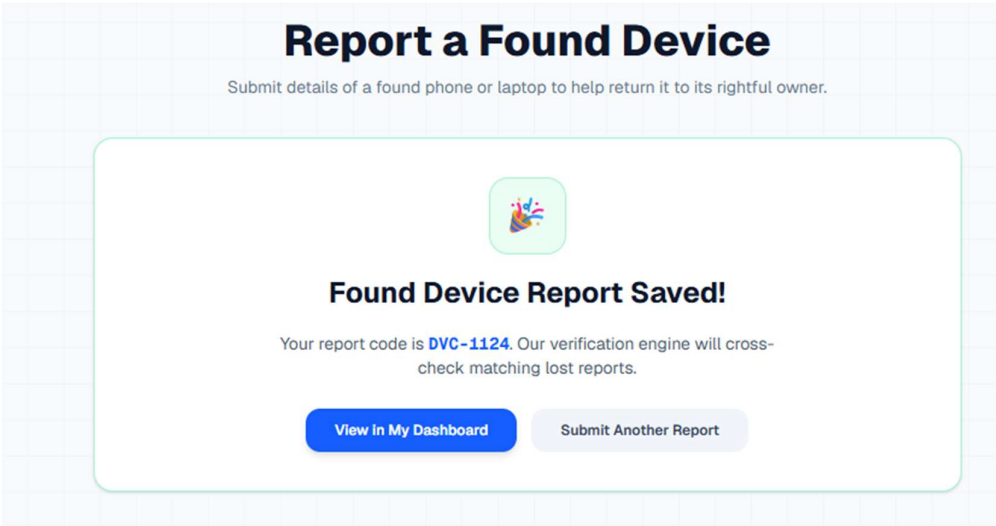


Fig. 6.5 Submitted Found Device Reporting Result

6.2 Verification Results

The Verification Results show that the TraceBack system supports device checking through IMEI/serial number and report code. When a user submits a verification value, the Laravel backend searches the device_reports table for matching records and returns the number of matches together with the corresponding device information. The verification function helps identify whether a reported device already exists in the system. If a matching IMEI, serial number, or report code is found, the relevant record is returned for further review. If no matching record exists, the system returns an empty result set. When no verification value is provided, the backend returns a validation message indicating that the required parameter is missing. It is important to note that a successful database match does not automatically confirm ownership or final recovery. Final verification remains under administrative control. Administrators can review the report and update its status to verified or returned when the necessary verification process has been completed. The verification results are summarized in Table 6.2.

Verification Function	**Result**
IMEI Search	Matching device records can be identified.
Serial Number Search	Stored serial-based records can be retrieved.
Report Code Search	Matching DVC-XXXX report can be returned.
No-Match Condition	No matching records are returned.
Empty Verification Input	Validation message is returned.
Match Count	Number of matching records is included in the response.
Final Verification	Controlled through administrative review
Verified/Returned Status	Verification state is updated accordingly.
Pre-Approval State	Matching reports remain Under Review and undisclosed.
Post-Approval Dashboard	Verified match, comparison, and handover information becomes available to the owner.

Table 6.2 Verification Results of the TraceBack System

The verification results indicate that the system provides a structured method for identifying potentially matching lost and found device records while keeping the final verification process under administrative control. This reduces the possibility of treating a database match as automatic proof of ownership or recovery. The implemented result therefore separates technical matching from authorized recovery. Matching IMEI or serial-number records become actionable to the owner only after administrative approval, at which point the dashboard presents the Admin Verified Match banner, status badge, side-by-side report details, handover guidance, verified contact desk, and claim-slip printing facility.

6.2.1 Device Verification Search Result

The implemented Device Verification Search function allows users to check whether a device record exists in the TraceBack system using an IMEI, Serial Number, or Report Code. When a verification value is submitted, the Laravel backend searches the

device_reports table for matching records and returns the corresponding device information. If a matching record is found, the system displays the available device details and current report status. If no matching record exists, the system returns a no-match result. When the verification field is left empty, an appropriate validation message is displayed. These results demonstrate that the TraceBack system can successfully search and identify stored device records using device identifiers or report codes. However, a successful search result represents a database match only and does not automatically confirm ownership or final recovery, which remains subject to administrative verification. The Device Verification Search result is illustrated in Fig. 6.6.

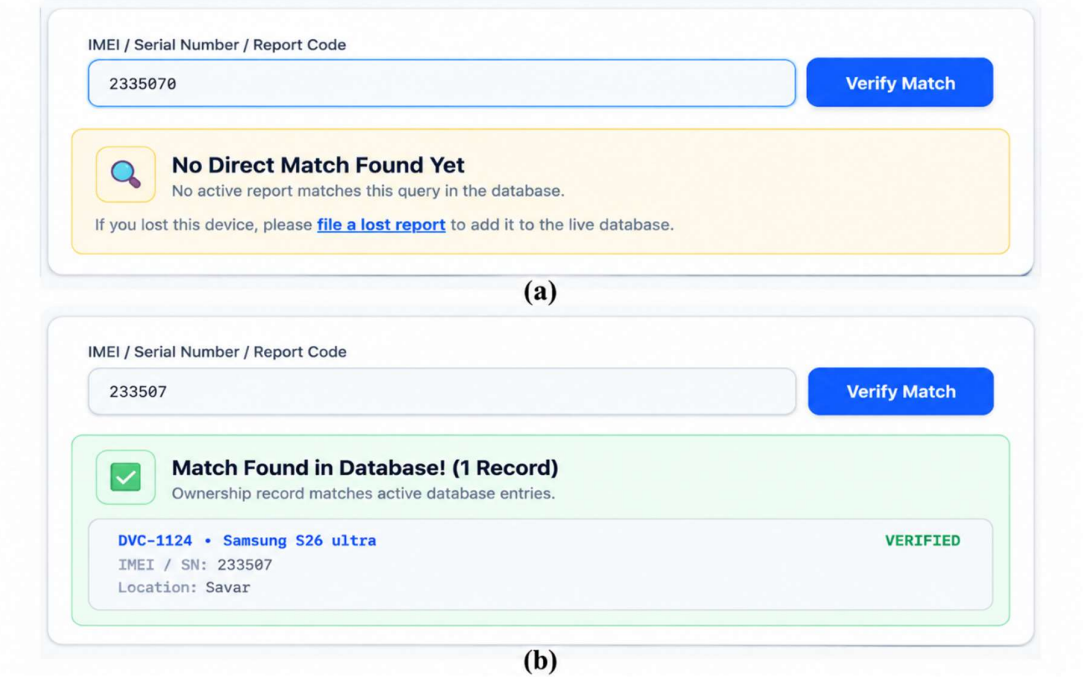


Fig. 6.6 Device Verification Search Results: (a) No Direct Match Found, and (b) Match Found in Database

6.2.2 Lost and Found Device Matching Result

The Lost and Found Device Matching function successfully identifies potential matches between lost-device and found-device reports stored in the TraceBack system. When the same IMEI or Serial Number is detected in corresponding lost and found reports, the system treats the records as a potential match and keeps them under administrative review. After the administrator verifies the device information and supporting evidence, the match can be approved and the relevant status is updated to matched or verified. Following approval, the rightful owner can view an Admin-Verified Device Match notification containing the lost-device information, matched found-device information, found location, report date, and handover instructions. The system also provides verified contact information and a Print Claim Slip option to support the recovery process. Sensitive finder information and identity documents remain restricted from ordinary users. These results demonstrate that the TraceBack system successfully separates automatic database matching from final

administrative approval and supports a controlled lost-and-found device recovery process. The successful matching result is illustrated in Fig. 6.7.

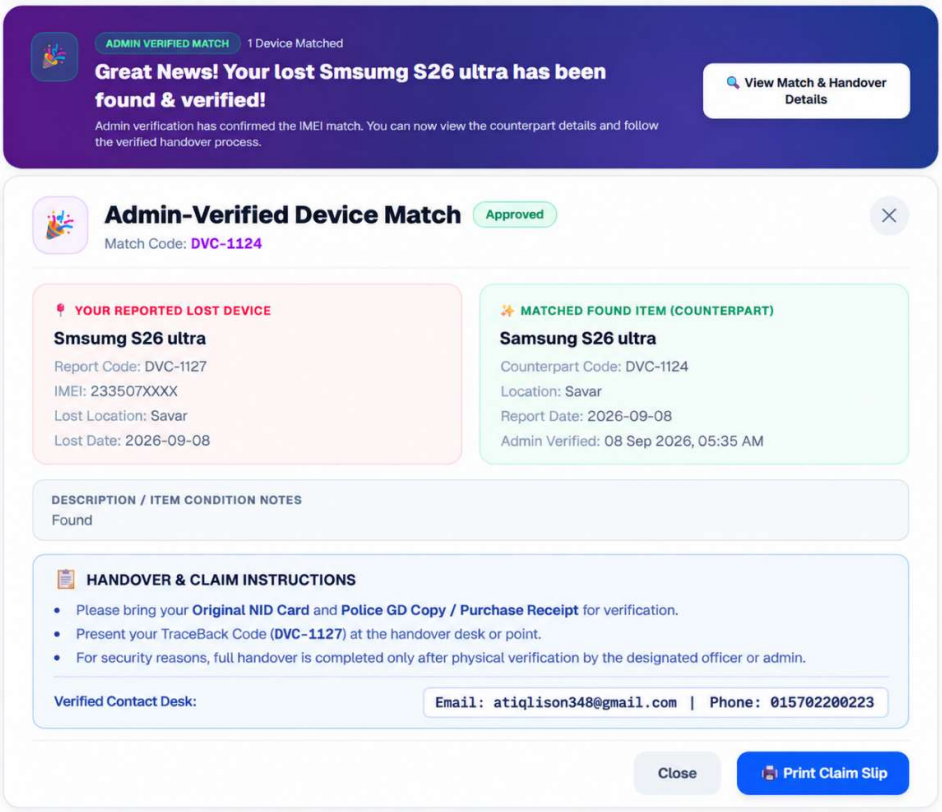


Fig. 6.7 Lost and Found Device Matching Result

6.3 Administrative Results

The Administrative Results demonstrate that the TraceBack system provides a centralized administrative interface for monitoring and managing lost and found device reports. The admin dashboard presents important system information, including total reports, recovered devices, pending claims, and registered users, together with an active verification queue for recently submitted reports. Administrators can review individual device records containing the report code, device information, IMEI or serial number, location, report type, and current status. The system also allows administrators to update a report to pending, matched, verified, returned, or flagged. When a report is changed to verified or returned, the corresponding verified field is automatically set to true. The administrative functions also support report searching, filtering, approval, and deletion. In addition, the admin interface provides access to device reports, user-related information, proof records, and system settings through a common navigation structure. The administrative results are summarized in Table 6.3.

Administrative Function	Result
Dashboard Monitoring	System summary information is displayed.
Verification Queue	Recent reports are available for review.
Report Search and Filtering	Relevant records can be located efficiently.
Status Management	Report status can be updated.
Report Verification	Verified/returned reports update the verification state.
Report Approval	Reports can be approved through the admin interface.
Report Deletion	Selected reports can be removed.
User Management	User-related information can be accessed.
Candidate Match Review	Lost and found records can be compared before approval.
Privacy-Controlled Handover	Sensitive finder documents remain admin-only while approved owner instructions are released.

Table 6.3 Administrative Results of the TraceBack System

The administrative results indicate that the implemented admin panel provides the necessary controls for monitoring device reports, managing verification activities, and maintaining report status within the TraceBack system. This centralized management structure supports consistent administrative handling of lost and found device records. The administrator-approval workflow further strengthens this control by keeping unapproved matches in the Under Review state and limiting sensitive finder documents to the administrative interface. Approved matched or verified records activate only the recovery information required by the rightful owner.

6.3.1 Administrative Report Review

The Administrative Report Review function allows authorized administrators to examine submitted lost- and found-device reports through the TraceBack admin panel. The administrator can review important report information, including the report code, device category, brand, model, IMEI or Serial Number, location, report type, current status, and available proof documents. The system also supports searching and filtering reports to help administrators locate relevant records efficiently. During the review process, the administrator can compare submitted information with the available ownership or supporting evidence and decide whether a report should remain pending, be marked as matched, verified, returned, or flagged. Invalid or unnecessary reports can also be removed when required. These results demonstrate that the administrative review function provides centralized control for monitoring submitted reports, checking supporting information, and managing report status before final verification or handover. The Administrative Report Review result is illustrated in Fig. 6.8.

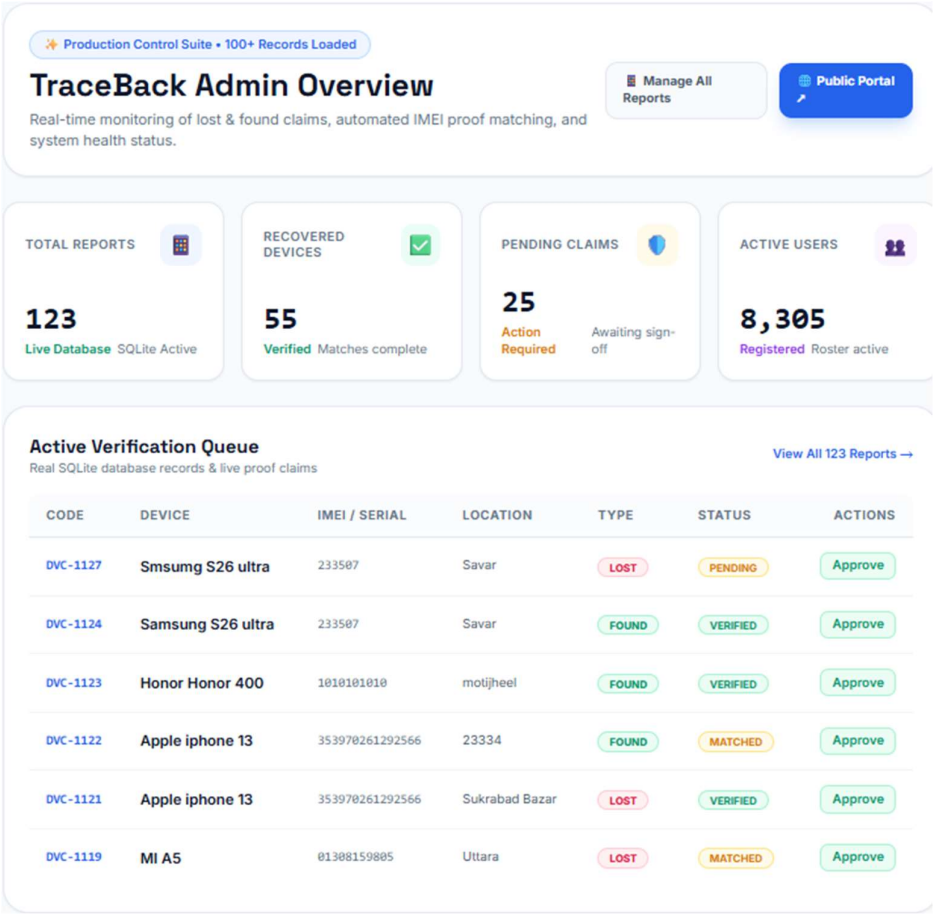


Fig. 6.8 Administrative Report Review Result

6.3.2 Admin-Verified Match and Handover Information

The Admin-Verified Match and Handover function presents recovery information to the rightful owner only after a potential lost-and-found device match has been reviewed and approved by an administrator. Before approval, matching reports remain in the Under Review state and the corresponding found-device and handover information is not disclosed to the user. After the administrator confirms the match and changes the report status to matched or verified, the owner's dashboard displays an Admin Verified Match notification. The user can then access the matched lost- and found-device information, found location and reporting date, item-condition notes, approved handover instructions, and verified contact-desk information. The system also provides a Print Claim Slip option to support the device recovery process. Sensitive finder information, including identity documents, remains restricted to authorized administrators. These results demonstrate that the TraceBack system provides a controlled and privacy-aware handover process after administrative verification. The Admin-Verified Match and Handover Information result is illustrated in Fig. 6.9.

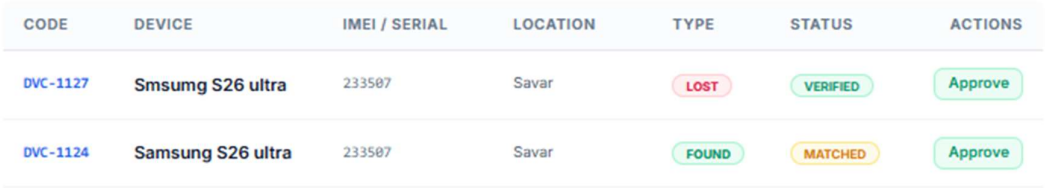


Fig. 6.9 Admin-Verified Match and Handover Information

6.4 Usability and System Organization

The TraceBack Lost and Found System is organized to provide a clear separation between public user functions and administrative functions. The frontend contains dedicated pages for lost-device reporting, found-device reporting, device searching, verification, and other user activities. This organization helps users access each function through a specific and structured interface. The reporting interfaces collect important information such as device brand, model, category, IMEI or serial number, location, contact information, and supporting documents through organized input fields. Search and verification functions are also maintained separately from report submission, which helps keep the overall workflow understandable and logically arranged. From a technical perspective, the system separates the frontend, backend, database, administrative interface, and file-storage components. The Next.js frontend manages user interaction, while the Laravel backend processes requests, performs validation, communicates with the SQLite database, and handles proof-document storage. Laravel controllers and Eloquent models further organize request processing and database operations. The administrative section is also structured through separate navigation options for the dashboard, device reports, user-related information, proof review, and system settings. This separation keeps administrative functions independent from the public-facing features of the system. Overall, the organization of the TraceBack system provides a clear and maintainable structure for both users and administrators. However, no formal usability survey or user-experience evaluation was included in the available project materials; therefore, this discussion is based on the implemented interface and system organization rather than measured usability scores. Within the authenticated user dashboard, approved matches are made easy to recognize through a dedicated alert banner, status badge, Match Info modal, and Print Claim Slip action. The modal organizes the lost and found records side by side and presents collection instructions without exposing the finder's sensitive identity evidence [21], [22], [23], [30].

6.5 Limitations

Although the TraceBack Lost and Found System implements the major functions required for device reporting, searching, verification, proof handling, and administrative management, the current implementation has several limitations. First, the device verification mechanism mainly depends on searching stored IMEI/Serial Numbers and report codes [6], [7]. The current system performs identifier-based matching using the available database records but does not include advanced intelligent matching techniques based on device images, multiple attributes, or machine learning-based similarity analysis. Second, the current database structure does not maintain a direct relationship between individual device reports and registered user accounts through a user_id foreign key. As a result, device reports and user information are maintained as separate entities, which can limit personalized report tracking and user-based management. Third, proof documents such as box memo, GD copy, NID proof, and owner photograph are optional during report submission. Although the system supports proof-document handling, missing documents may reduce the available evidence for verification. Therefore, future versions can improve verification reliability by introducing mandatory proof requirements or advanced document verification mechanisms. Another limitation is that the system currently operates only with its internal database and does not directly integrate with external databases or services such as BTRC, NEIR, GSMA, manufacturers, telecom operators, or law-enforcement systems. Finally, the final verification decision still depends on administrative review. A matching device record does not automatically confirm ownership or recovery. Future improvements can introduce AI-based matching, automated verification support, and stronger authentication mechanisms to improve accuracy and scalability.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

The TraceBack Lost and Found System was developed as a web-based platform for reporting, searching, verifying, and managing lost and found electronic devices. It combines a Next.js frontend, Laravel 12 backend, SQLite database, proof-document storage, device verification functions, and an administrative management interface. Users can submit lost or found device reports with relevant device information and supporting documents. The system validates each submission, generates a unique report code, assigns an initial pending status, and securely stores the report. TraceBack also enables device searching and verification using IMEI or serial number, report code, brand, model, and location. Administrators can review reports and ownership evidence, monitor verification activities, update report status, and manage invalid records. Testing confirmed the major functions of the system, including report submission, searching, verification, status management, administrative operations, and proof-file validation. A major contribution of the system is its administrator-approved matching and handover workflow. Potential lost-and-found matches remain Under Review until an administrator verifies and approves them. After approval, the rightful owner receives a verified-match notification, matched-device information, handover instructions, verified contact details, and a printable claim slip. Sensitive finder information remains restricted to administrators, improving privacy and reducing fraudulent or premature claims.

7.2 Future Improvements

Future versions of TraceBack may introduce AI-assisted matching with confidence scores, OCR-based identifier extraction, image comparison, location-based matching, and fraud or duplicate-claim detection. Stronger authentication, secure cloud storage, integration with trusted external verification services, and migration to MySQL or PostgreSQL could further improve the system’s accuracy, privacy, reliability, and scalability.

REFERENCES

[1] International Telecommunication Union, *Framework for Combating the Use of Stolen Mobile Devices*, ITU-T Recommendation Q.5051, Geneva, Switzerland, Mar. 2020. [Online]. Available: https://www.itu.int/rec/T-REC-Q.5051-202003-I/en. [Accessed: Sep. 7, 2026].

[2] GSM Association, "GSMA Device Check: Device Status Verification." [Online]. Available: https://devicecheck.gsma.com/. [Accessed: Sep. 7, 2026].

[3] Bangladesh Telecommunication Regulatory Commission, "National Equipment Identity Register (NEIR)." [Online]. Available: https://neir.btrc.gov.bd/. [Accessed: Sep. 7, 2026].

[4] OWASP Foundation, "File Upload Cheat Sheet," *OWASP Cheat Sheet Series*. [Online].Available: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html. [Accessed: Sep. 7, 2026].

[5] A. Taivalsaari, T. Mikkonen, C. Pautasso, and K. Systä, "Comparing the built-in application architecture models in the web browser," in *Proc. IEEE Int. Conf. Software Architecture (ICSA)*, 2017, pp. 51–54, doi: 10.1109/ICSA.2017.23.

[6] G. Farrell, "Preventing phone theft and robbery: The need for government action and international coordination," *Crime Science*, vol. 4, Art. no. 4, 2015, doi: 10.1186/s40163-014-0015-0.

[7] S. P. Rao, S. Holtmanns, I. Oliver, and T. Aura, "Unblocking stolen mobile devices using SS7-MAP vulnerabilities: Exploiting the relationship between IMEI and IMSI for EIR access," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 1171–1176, doi: 10.1109/Trustcom.2015.500.

[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.

[9] P. P. S. Chen, "The entity-relationship model—Toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, Mar. 1976, doi: 10.1145/320434.320440.

[10] S. Doğan, A. Betin-Can, and V. Garousi, "Web application testing: A systematic literature review," *Journal of Systems and Software*, vol. 91, pp. 174–201, May 2014, doi: 10.1016/j.jss.2014.01.010.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[12] R. Smith, “An overview of the Tesseract OCR engine,” in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633, doi: 10.1109/ICDAR.2007.4376991.

[13] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, Art. no. 15, July 2009, doi: 10.1145/1541880.1541882.

[14] W. G. J. Halfond, J. Viegas, and A. Orso, “A classification of SQL-injection attacks and countermeasures,” in *Proc. Int. Symp. Secure Software Engineering*, Arlington, VA, USA, 2006.

[15] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, “Current state of research on cross-site scripting (XSS): A systematic literature review,” *Information and Software Technology*, vol. 58, pp. 170–186, Feb. 2015, doi: 10.1016/j.infsof.2014.07.010.

[16] International Telecommunication Union, “Combat mobile device theft with blockchain-based global IMEI storage and services innovation,” presented at the *ITU-T Workshop on Security Aspects of Blockchain*, Geneva, Switzerland, Jul. 23, 2018. [Online]. Available: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/20180723/Documents/abstract.pdf. [Accessed: Sep. 7, 2026].

[17] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, doi: 10.1109/2.485845.

[18] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010, doi: 10.1145/1721654.1721672.

[19] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Boston, MA, USA: Addison-Wesley Professional, 2012.

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1994.

[21] J. Nielsen, *Usability Engineering*. Boston, MA, USA: Academic Press, 1993.

[22] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. Boston, MA, USA: Pearson, 2016.

[23] F. D. Davis, “Perceived usefulness, perceived ease of use, and user acceptance of information technology,” *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, Sep. 1989, doi: 10.2307/249008.

[24] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” in *Proc. IEEE Symp. Security and Privacy*, 2012, pp. 553–567, doi: 10.1109/SP.2012.44.

[25] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939.

[26] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, Oct. 2002, doi: 10.1142/S0218488502001648.

[27] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, Nov.–Dec. 2001, doi: 10.1109/69.971193.

[28] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, “Why Eve and Mallory love Android: An analysis of Android SSL (in)security,” in *Proc. ACM Conf. Computer and Communications Security (CCS)*, 2012, pp. 50–61, doi: 10.1145/2382196.2382205.

[29] D. Zissis and D. Lekkas, “Addressing cloud computing security issues,” *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, Mar. 2012, doi: 10.1016/j.future.2010.12.006.

[30] W. H. DeLone and E. R. McLean, “The DeLone and McLean model of information systems success: A ten-year update,” *Journal of Management Information Systems*, vol. 19, no. 4, pp. 9–30, 2003, doi: 10.1080/07421222.2003.11045748.

[31] P. Ammann and J. Offutt, *Introduction to Software Testing*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2016.

[32] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Boston, MA, USA: Pearson, 2016.