Droid Scanner: App Security Management System

SE- 505: Software Project Lab - 2

Submitted by

Md. Shaikhul Islam(BSSE 1438)
Salsabila Zaman (BSSE 1443)

Supervised by

Prof. Md. Zulfiquar Hafiz
Institute of Information Technology
University of Dhaka

Supervisor's Approval: _____

Date: 09-12-2024

Table of Contents

List Of Figures

Software Requirement Specification report of "Droid Scanner: App Security Management System"

User Story :

The **Droid Scanner: App Security Management System** is designed as a robust software solution, safeguarding android devices from potential threats and ensuring high-security standards for app development.

This system serves two primary stakeholder groups: **Android Device Users**, who prioritize device security, and **Android App Developers**, who need tools to ensure their apps are free from vulnerabilities and comply with privacy standards.

The software provides flexible scan options, offering a **Full Scan** : A comprehensive scan of all the installed apps and a **Quick Scan** : scan a particular App on command. All the forms ensure security monitoring.

For Android Device Users, the system operates as a personal security tool. Users initiate the setup by accessing their dedicated User Dashboard, which displays the user profile, previous scans history, and black & white applists. The system's Malicious App Detection feature performs in-depth scans to identify harmful apps by analyzing APK files. The APK Extraction feature allows users to save or examine APK files. With the Custom App Management feature, users can whitelist trusted apps to avoid redundant checks and blacklist risky apps. All scans and their outcomes are stored securely in a User Database, which enables easy access to past reports, scan summaries, helping users stay informed about potential risks and ensure data privacy.

For Android App Developers, the system serves as an essential security and compliance tool to evaluate and improve app safety before launch. Developers access a dashboard that enables them to view scan reports. The system's APK Extraction and Permissions tools let developers save and review copies of their apps, making sure permissions like location or camera are only used when necessary. The Permission Viewer shows every permission request in detail, helping developers

easily choose which to keep or remove for better user privacy. After each scan, the system generates a comprehensive report detailing **app details, permissions used, intents, security status, feature analysis, helping developers align their app with industry standards**. Additionally, each developer has access to a secure User Database where all detected issues, past scans, and security records are stored, ensuring a traceable history for continuous improvement and compliance.

Through the App Security Management System, both primary users and developers receive vital, tailored software for device protection and app security. The platform's streamlined dashboards, customizable scan options, in-depth reports, and centralized database empower users to maintain data integrity and developers to release trustworthy, compliant applications that prioritize user security and privacy.

QFD (Quality Function Deployment):

QFD, or Quality Function Deployment, is a structured approach used in product development and project management. It helps ensure that customer needs and expectations are translated into specific product or service features. Essentially, it's a way to bridge the gap between what customers want and what a company delivers. QFD involves capturing customer requirements, prioritizing them, and then aligning internal processes to meet those requirements efficiently. It's like a roadmap for turning customer desires into tangible results.

Normal:

1. The application will feature a user-friendly dashboard that consolidates all functionalities for ease of access.
2. Users will have the ability to update and edit their profiles.
3. The system will enable users to analyze APK files to extract the features utilized by the respective applications.
4. Users can review the history of their previous scans

Expected:

1. APK file can be uploaded
2. Malicious apps would get detected

Exciting:

1. Multiple Scan Options: Full scan for scanning all apps in the device, quick scan for a single or fewer apps.
2. Custom App Management: Whitelist for trusted apps, blacklist for risky ones, and this would be done by the choice of the users.

1. Use Case Diagram

A Use Case describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. In fact, a use case diagram is a kind of visualization of the system where an end-user has an idea of a specific feature. It simply describes a story using corresponding actors who perform important roles in the story and make the story understandable for the users.

The first step in writing a Use Case is to define the set of "actors" that will be involved in the story. Actors are the different people or systems that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as system operators. They procedure some information or consume some information. Every user has one or more goals when using the system.

Primary Actor

Primary actors interact directly to achieve the required system function and derive the intended benefit from the system. They work directly with the software. They produce some information and consume some information too.

Secondary Actor

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.
Here is the use case diagram to observe the non-technical view of the System.

Actors:

- Android Device Users
- Android App Developers

Primary Actor:

- Android Device Users
- Android App developers

Level 0:

Name: ASMS

Primary Actor: Android device users, Android app developers

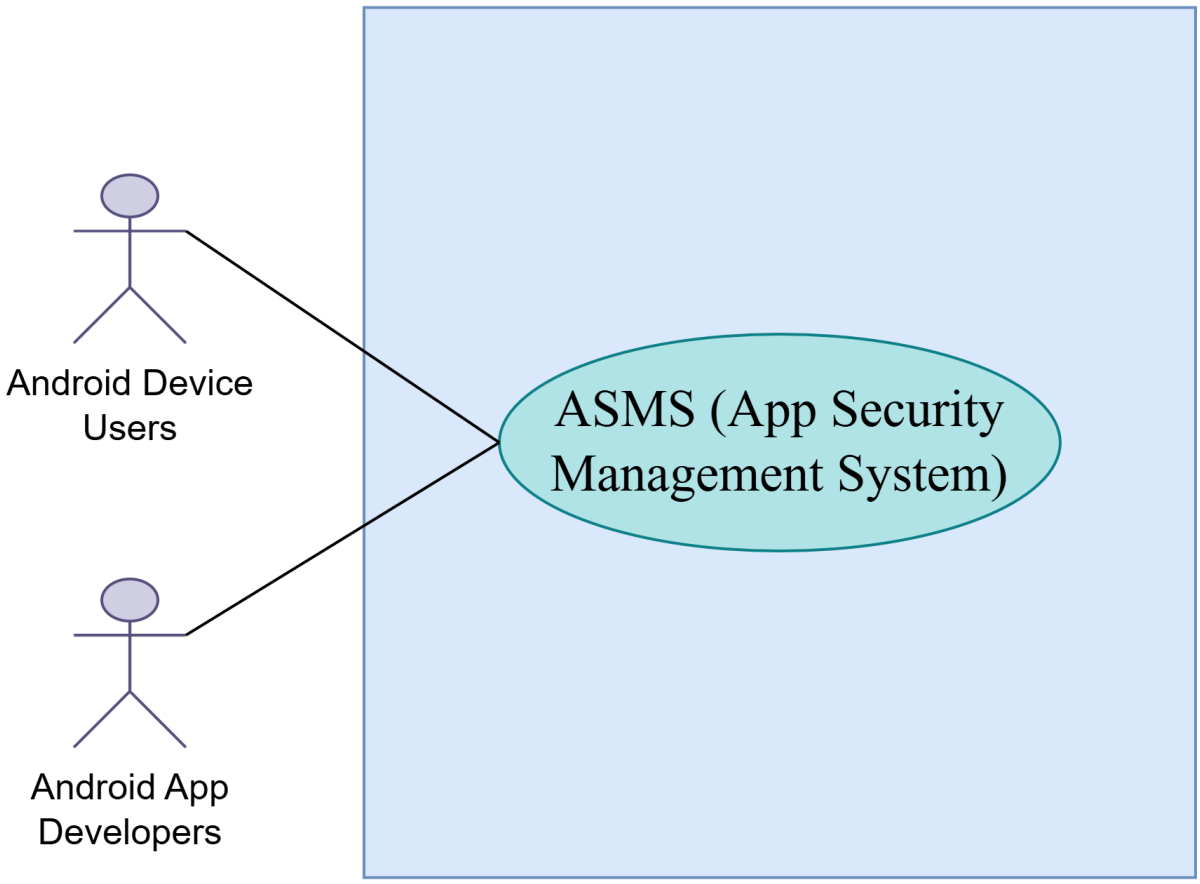

Figure 0

Level 1:

Name: ASMS(Detailed)

Primary Actor: Android device users, Android app developers

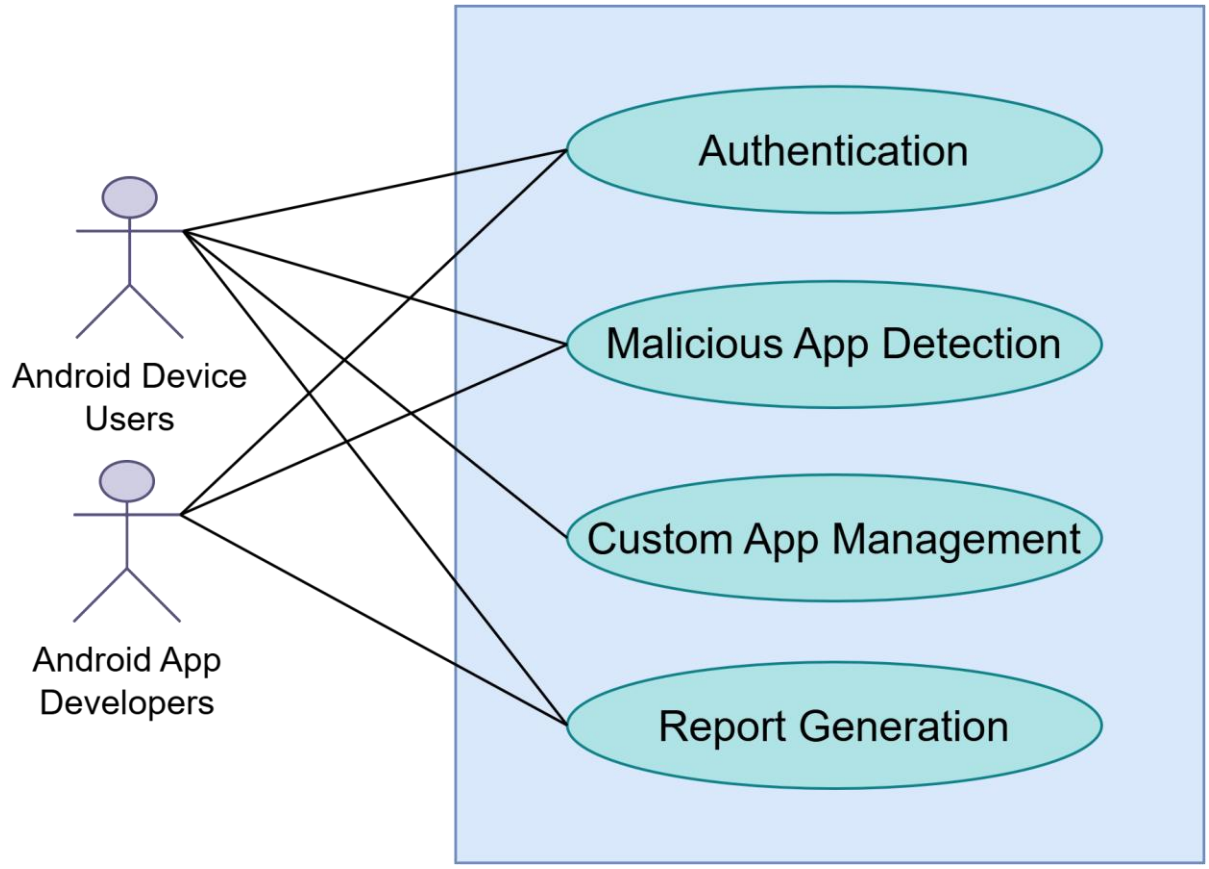

Figure 1

Level 1.1:

Name: Authentication

Primary Actor: Android device users, Android app developers

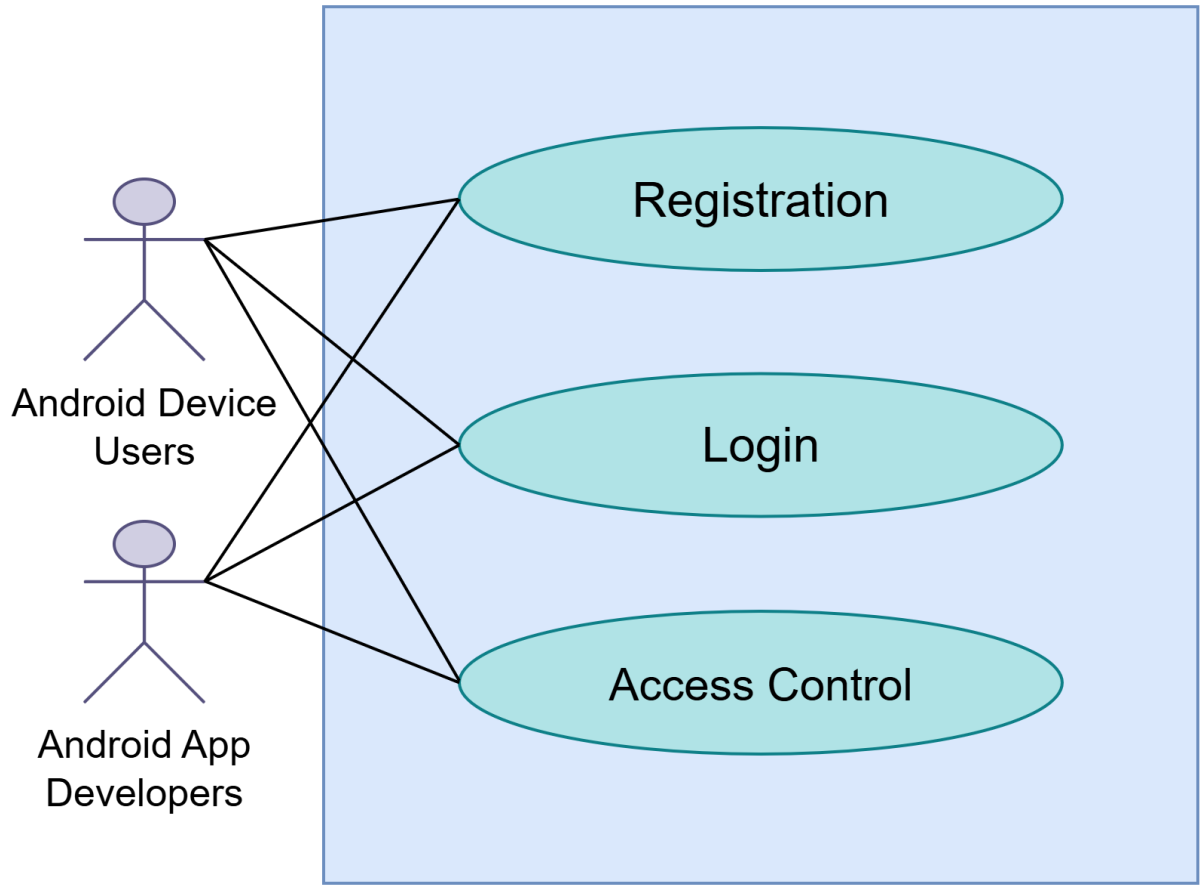

Figure 1.1

Level 1.2

Name: Malicious App Detection

Primary Actor: Android device users, Android app developers

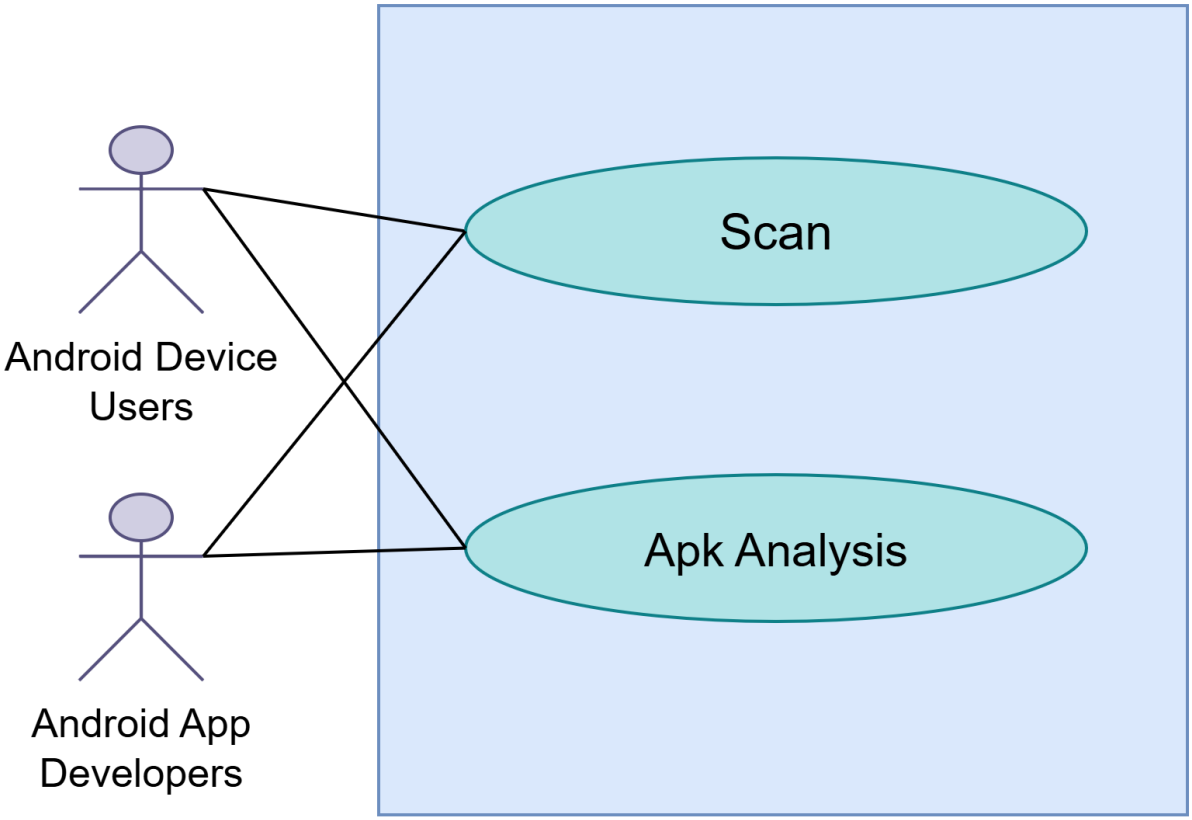

Figure 1.2

Level 1.2.1

Name: Scan

Primary Actor: Android device users, Android app developers

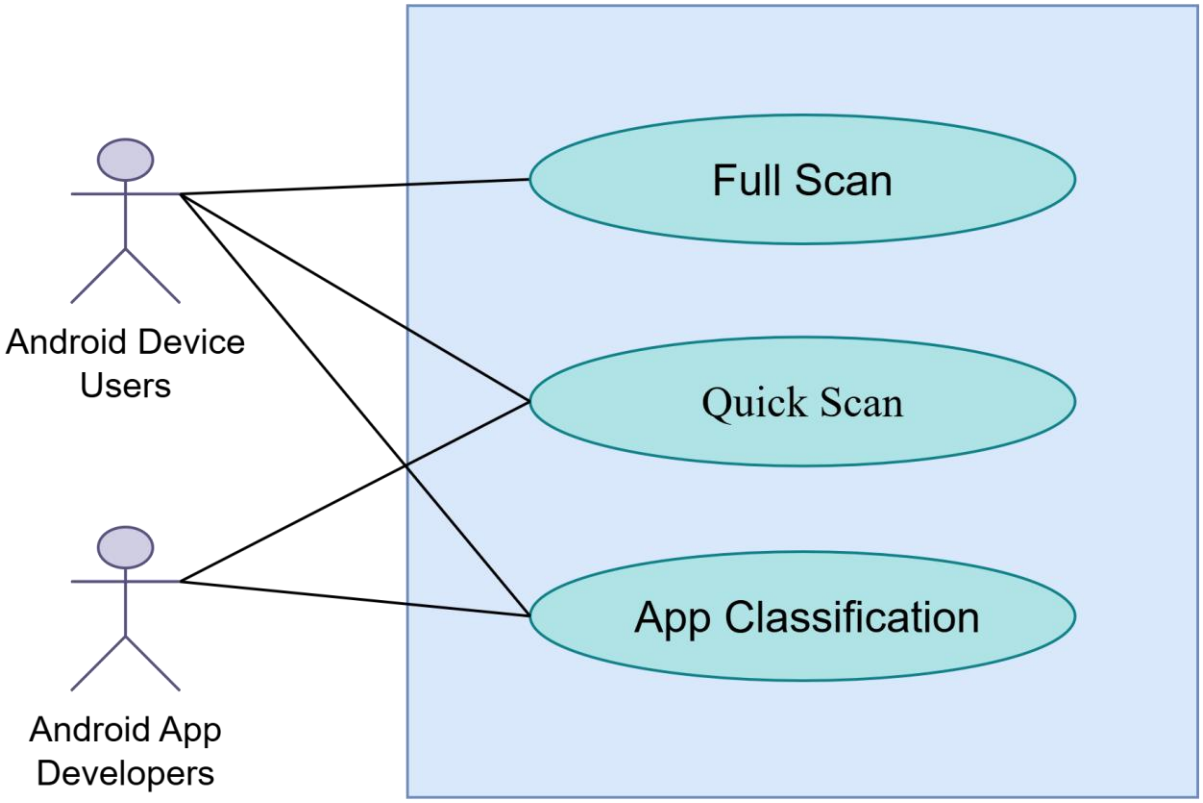

Figure 1.2.1

Level 1.2.2:

Name: APK Analysis

Secondary Actor: Android device users, Android app developers

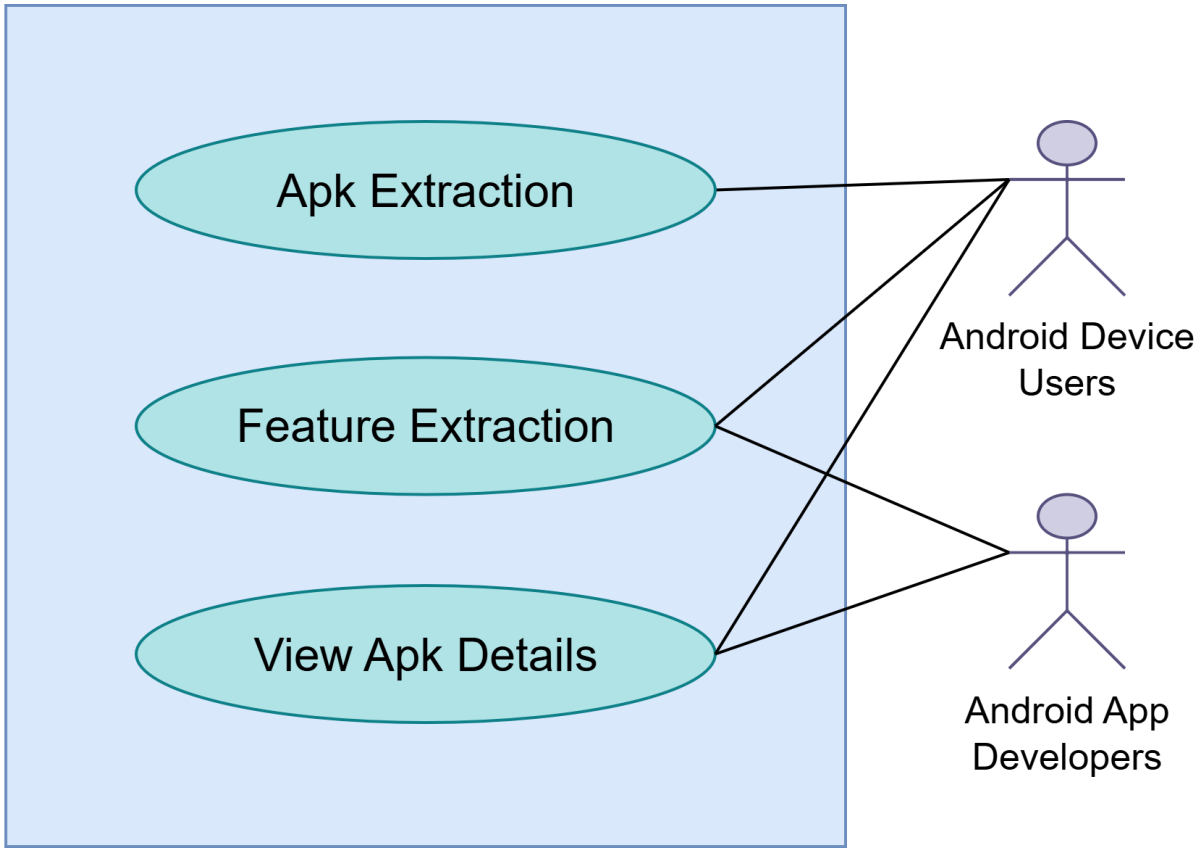

Figure 1.2.2

Level 1.3

Name: Custom App Management

Primary Actor: Android device users

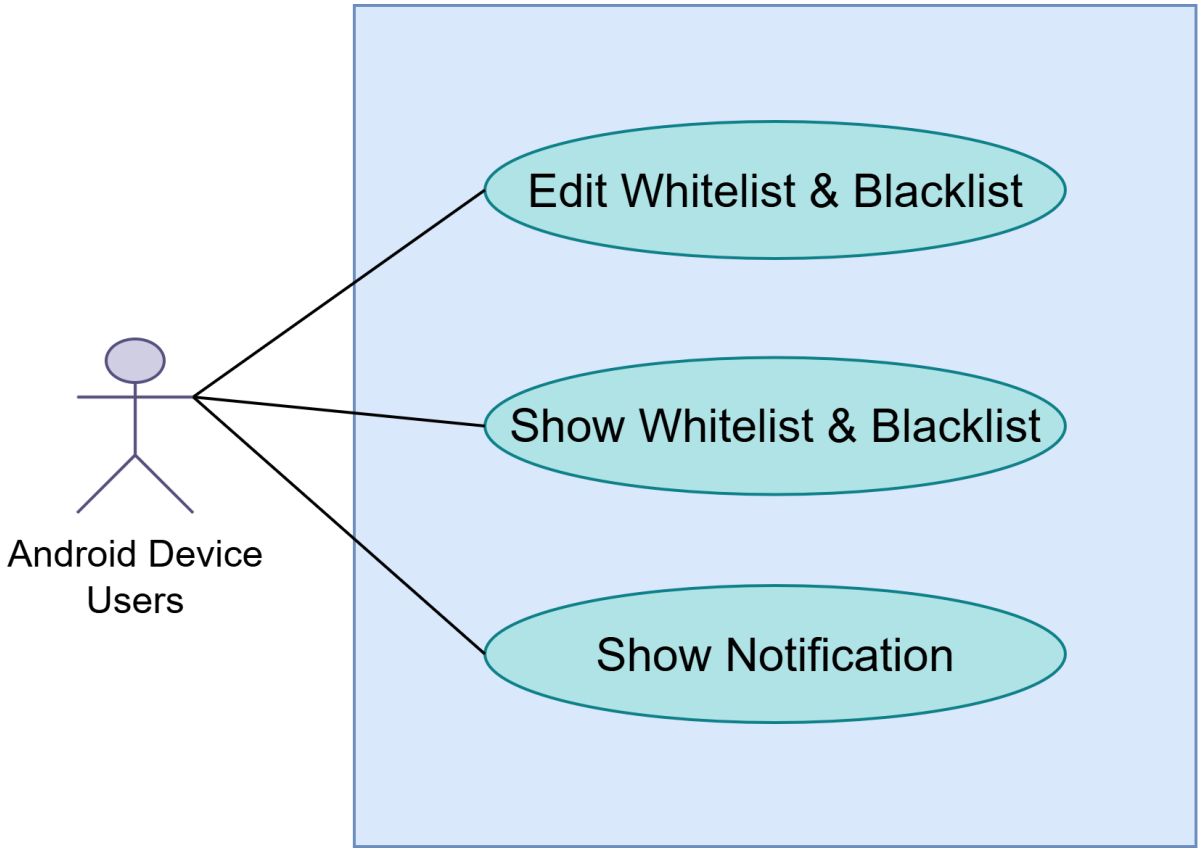

Figure 1.3

Level 1.4

Name: Report Generation

Secondary Actor: Android device users, Android app developers

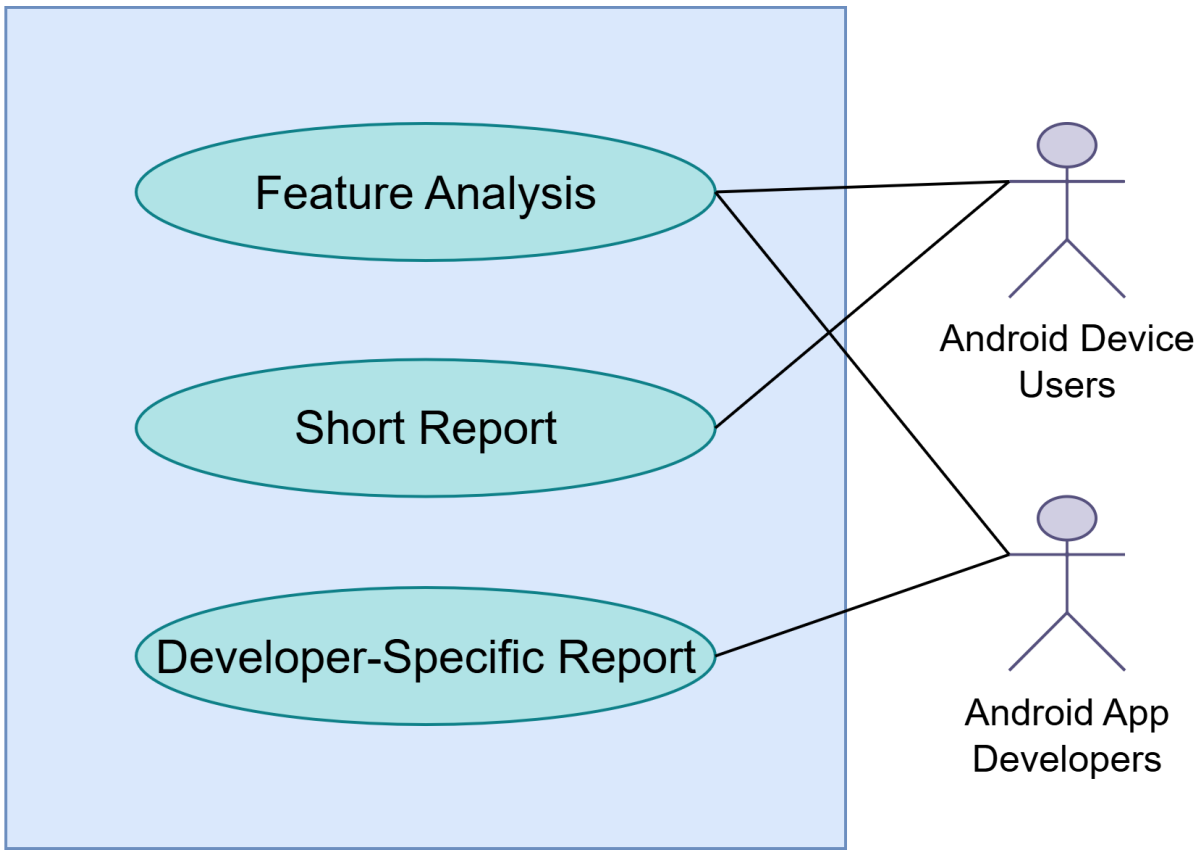

Figure 1.4

2.Swimlane Diagram

2.1. Definition of Swimlane Diagram

If software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated, then the software team chooses to create data models as part of overall requirements modeling. The entity-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects, and the information about how the data objects are entered, stored, transformed, and produced within the system.

2.2. Swimlane ID (SID) 1.1

Name: Authentication

Reference: Use Case Level 1.1

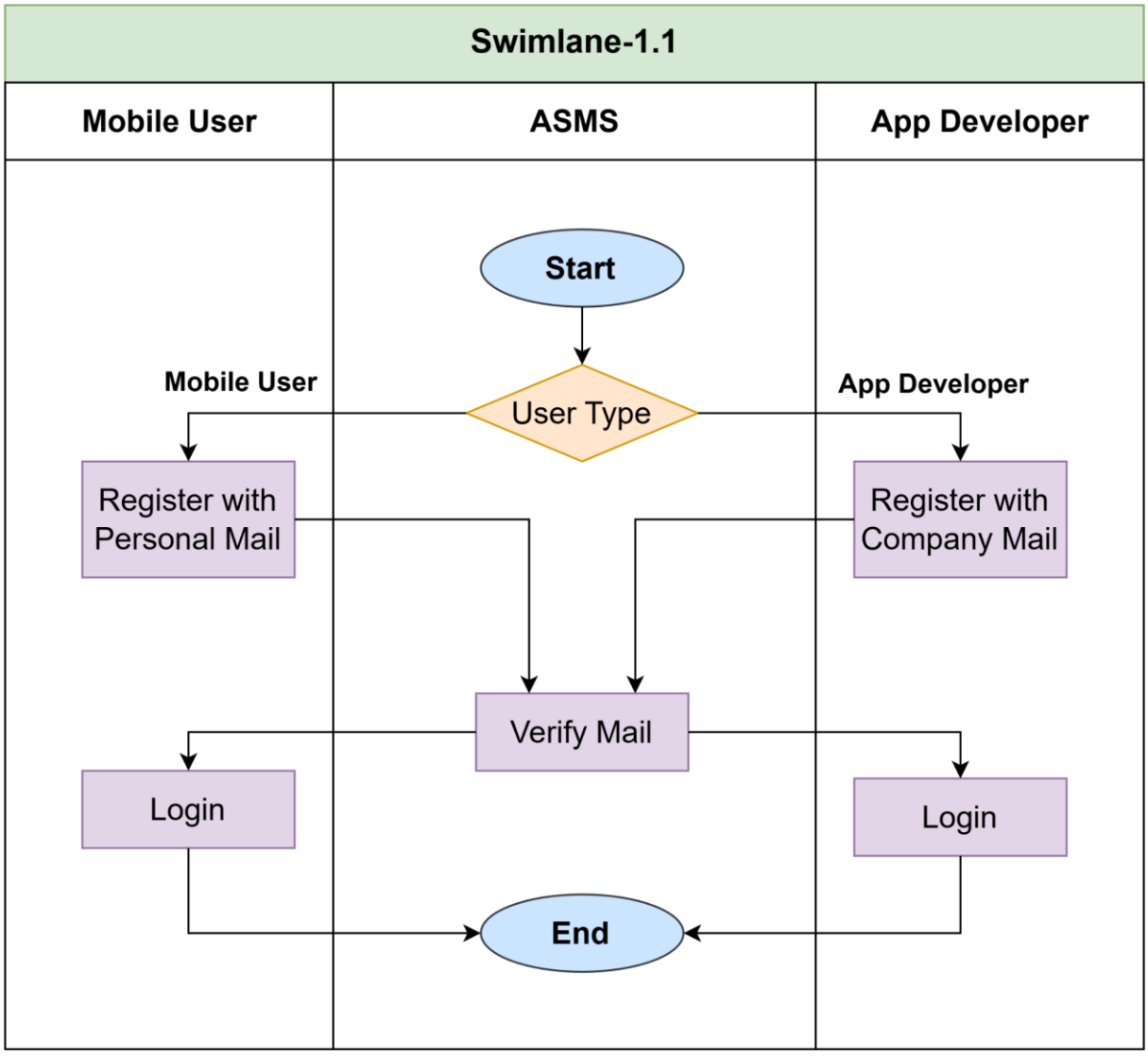

Figure: SID 1.1

2.3. Swimlane ID (SID) 1.2

Name: Malicious App Detection

Reference: Use Case Level 1.2

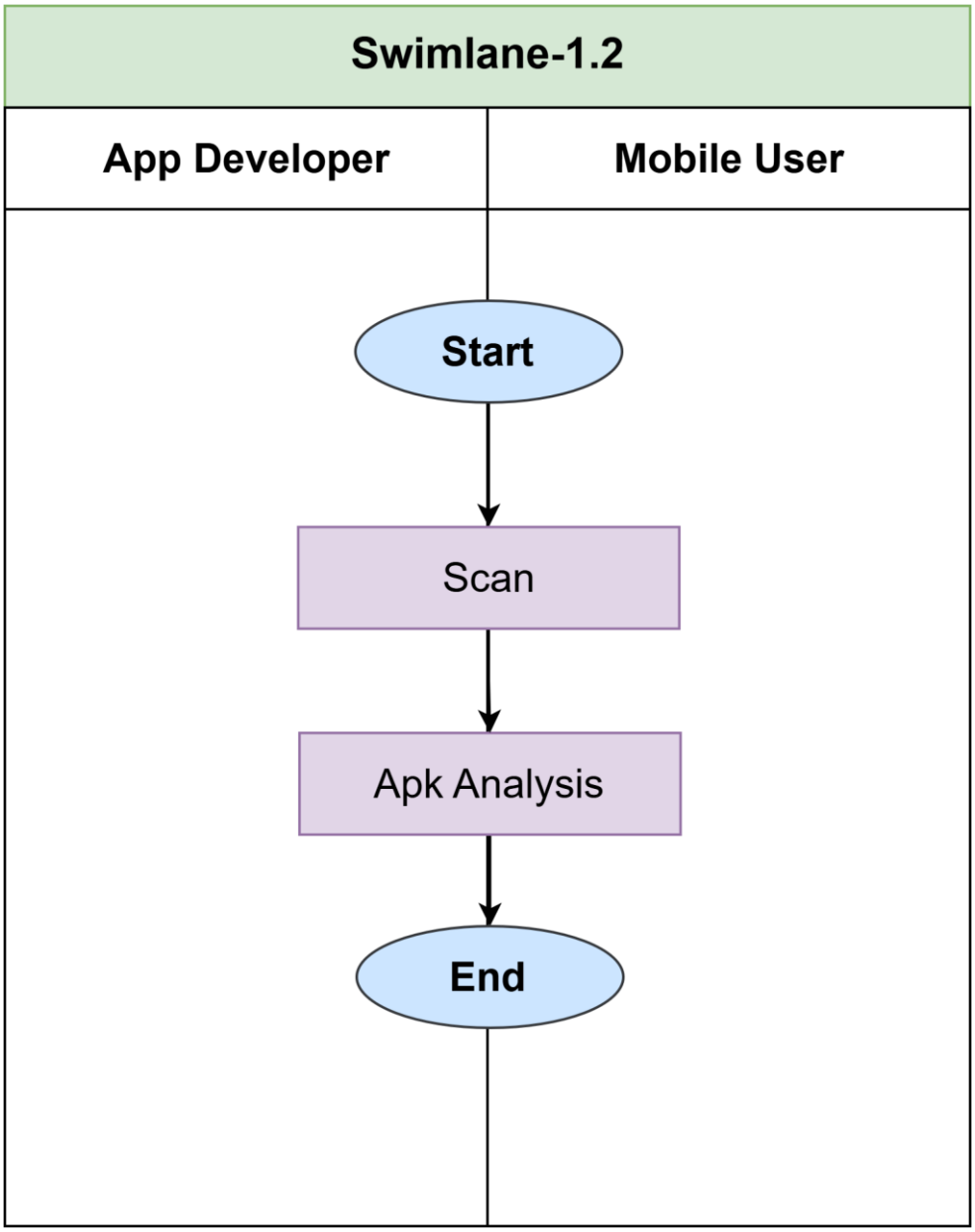

Figure: SID 1.2

2.3. Swimlane ID (SID) 1.2.1

Name: Scan

Reference: Use Case Level 1.2.1

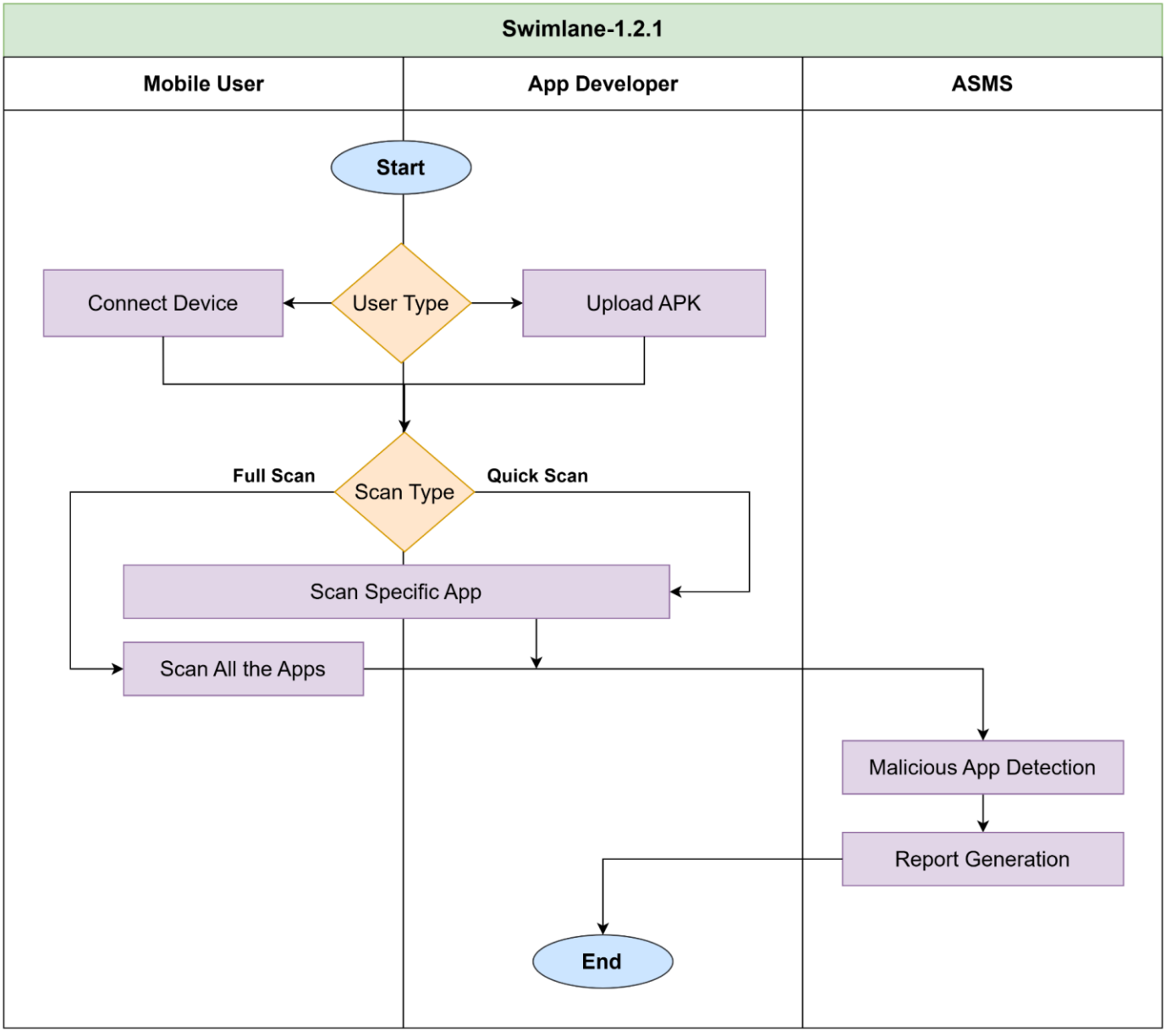

Figure: SID 1.2.1

2.4. Swimlane ID (SID) 1.2.2

Name: APK Analysis

Reference: Use Case Level 1.2.2

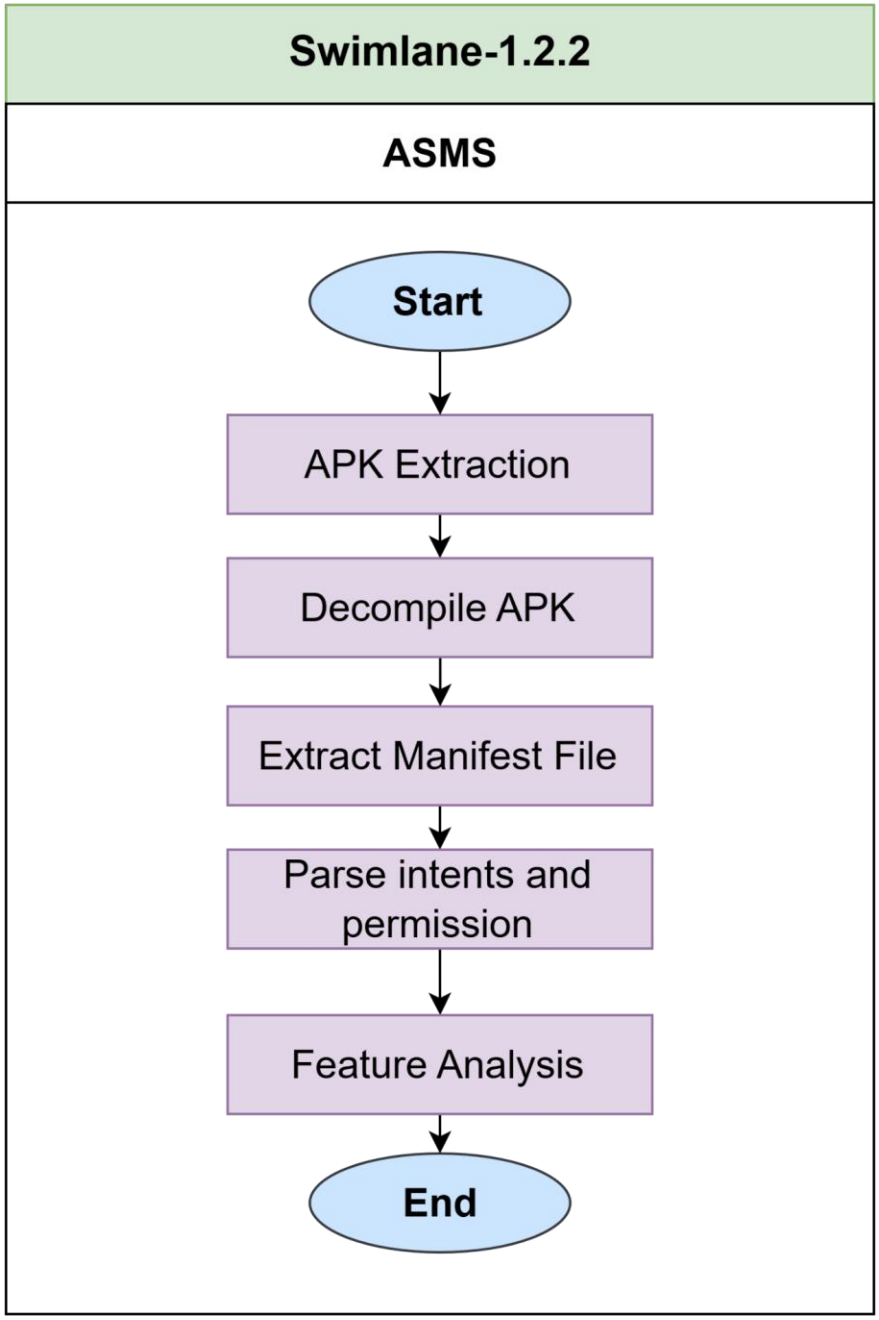

Figure: SID 1.2.2

2.5. Swimlane ID (SID) 1.3

Swimlane-1.3: Custom App Management

Reference: Use Case Level 1.3

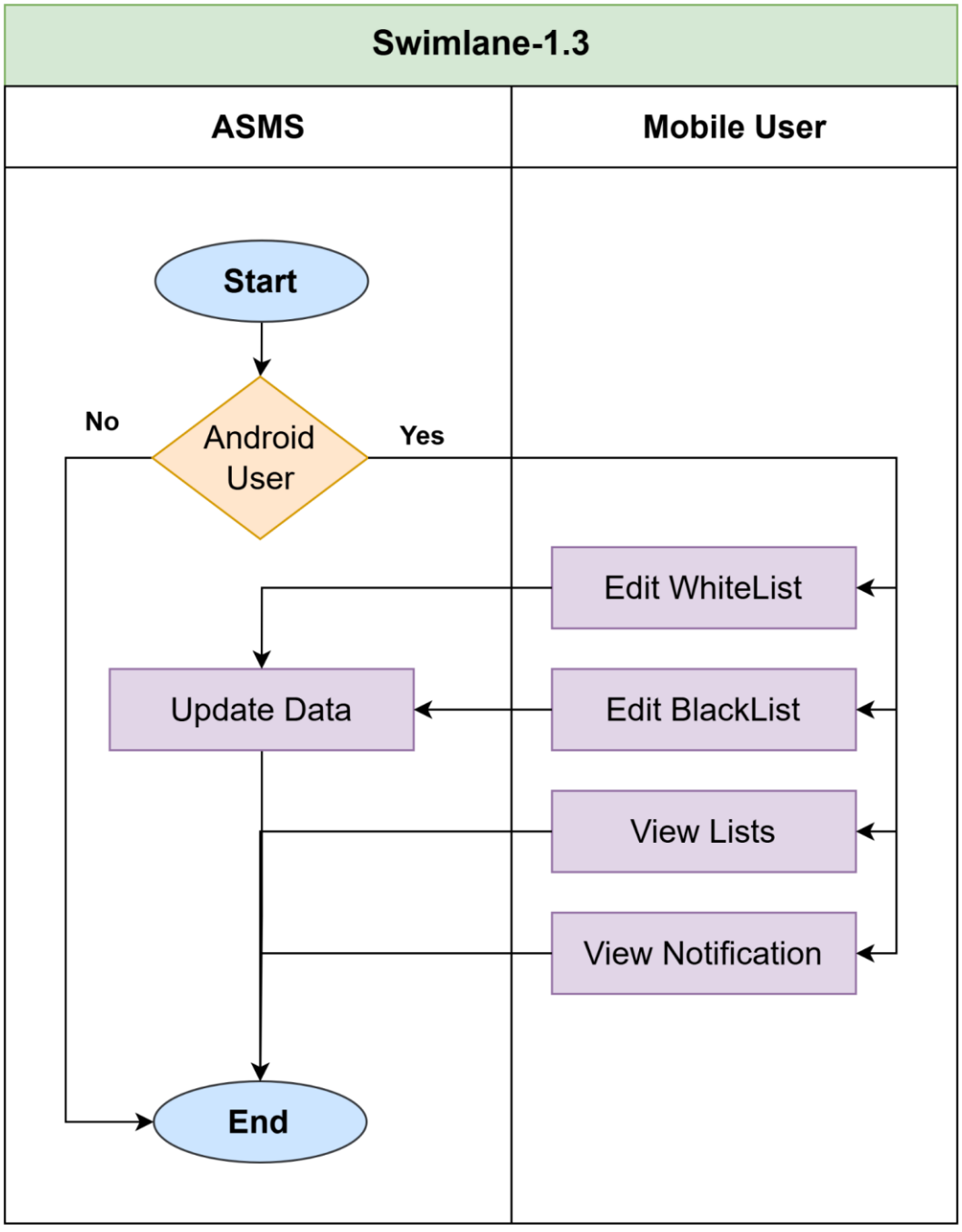

Figure: SID 1.3

2.6. Swimlane ID (SID) 1.4

Name: Report Generation

Reference: Use Case Level 1.4

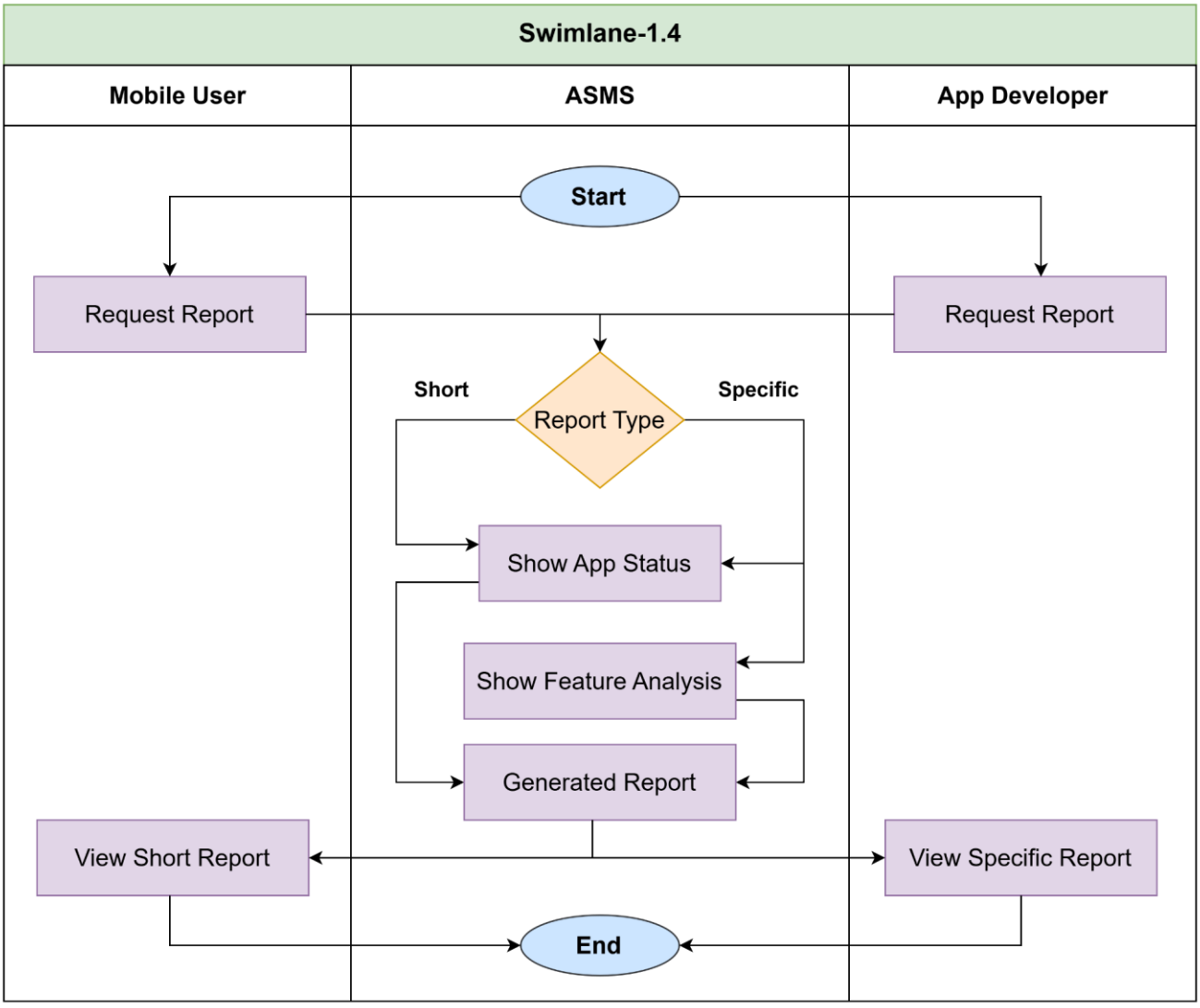

Figure: SID 1.4

3.Data Based Modeling

3.1. Definition of Database Diagram

Data modeling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations. Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system.

Data Objects

A data object is a representation of composite information that must be understood by the software.

3.2. Data Objects Identification:

SL	Noun	Problem/Solution Space	Attributes
1	Droid Scanner	S	
2	App	S	64, 9, 35, 51, 59, 62
3	Security	S	
4	Management System	S	
5	Software	S	
6	Solution	S	
7	Mobile Device	S	
8	Threat	P	
9	Name	S	

10	App Development	S	
11	User	S	64, 9, 28, 57, 27
12	Groups	S	
13	Primary Mobile Device Users	S	
14	Device Security	S	
15	Mobile App Developers	S	
16	Tools	S	
17	Vulnerabilities	P	
18	Privacy	S	
19	Options	S	
20	Full Scan	S	
21	Quick Scan	S	
22	Command	S	
23	Forms	S	
24	Security Monitoring	S	
25	Tool	S	
26	Setup	S	
27	User Dashboard	S	
28	Profile	S	
29	Scans History	S	
30	Blacklist	S	
31	Whitelist	S	
32	Applist	S	

33	Malicious App Detection	S	
34	Feature	S	
35	APK Files	S	
36	APK Extraction	S	
37	Custom App Management	S	
38	Notification	S	64, 11, 76, 74, 62
39	Risky Apps	S	
40	Scan	S	64, 74, 2, 50,
41	Outcome	S	
42	User Database	S	
43	Reports	S	64, 40, 58, 51, 69, 59, 60, 61
44	Scan Summaries	S	
45	Data Privacy	P	
46	Compliance	P	
47	Safety	P	
48	Launch	S	
49	Dashboard	S	64, 11, 28, 29, 30, 31
50	Scan Reports	S	
51	Permissions	S	
52	Location	S	
53	Camera	S	
54	Permission Viewer	S	
55	Request	S	

56	Privacy	P	
57	Role	S	
58	App Details	S	
59	Intents	S	
60	Feature Analysis	S	
61	API Calls	S	
62	Status	S	
63	Industry Standards	S	
64	ID	S	
65	Security Records	P	
66	App History	P	
67	Improvement	P	
68	Platform	S	
69	Security Status	S	
70	Database	S	
71	Integrity	S	
72	Applications	S	
73	Protection	P	
74	Date	S	
75	Result	S	
76	Message	S	

3.3. Final Data Objects:

1. **User**
    - Attributes: User_ID, Name, Role (Primary User/Developer), Profile
2. **App**
    - Attributes: Package_Name, Name, APK_File, Permissions, Intent, Status (Malicious/Benign), API_Calls,
3. **Scan**
    - Attributes: Scan_ID, Type (Full/Quick), Date, App_ID (FK), User_ID (FK), Scan_Results
4. **Report**
    - Attributes: Report_ID, Scan_ID (FK), App Details, Permissions Used, Intent Analysis, Security Status
5. **Notification**
    - Attributes: Notification_ID, User_ID (FK), Message, Date, Status (Read/Unread)

3.4. Relationship between Objects:

Data Object	Relationship	Related Data Object
Scan	has	Report
User	performs	Scan
User	gets	Notification
User	has	App
App	analyzed through	Scan
Notification	gets	Report

3.5. Entity Relation (ER) Diagram:

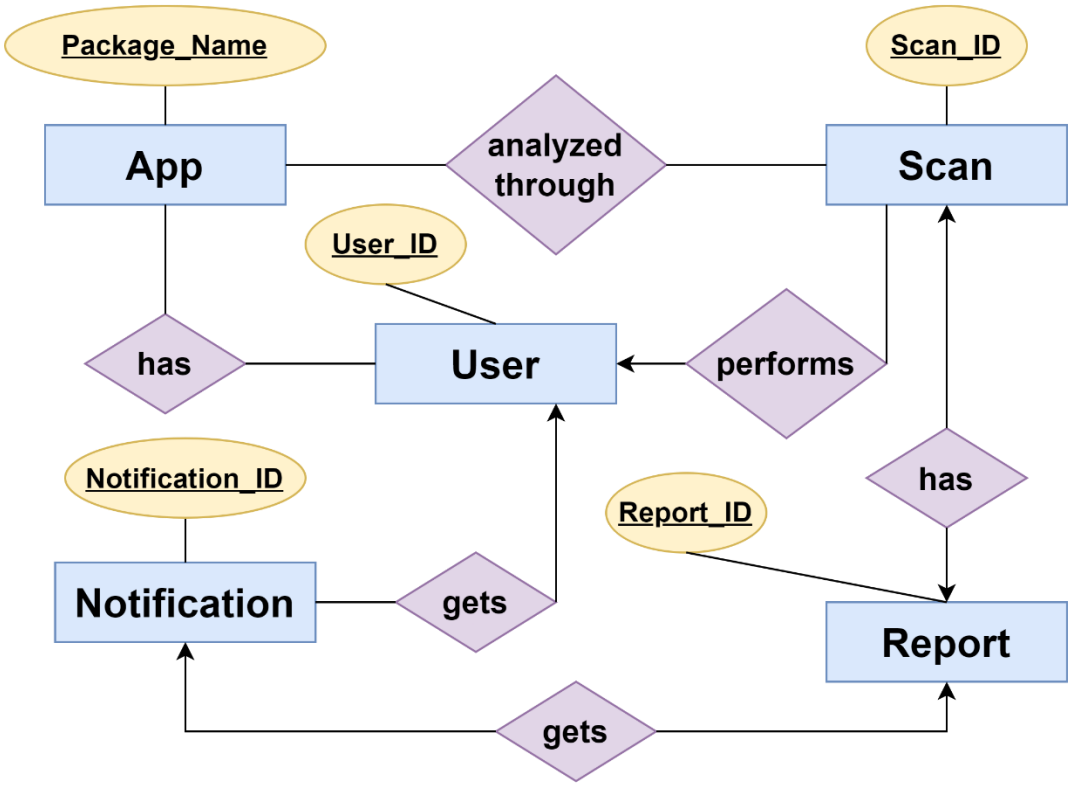

Figure 3

3.6. Schema Diagram:

Data Object	Attribute	Type	Size
User	User_ID Name Role Profile	Varchar Varchar Varchar Varchar	100 100 50 100
App	Package_Name Name Status APK_File Permissions Intent API_Calls Type	Varchar Varchar Varchar Blob Text Text Text Text	100 100 50
Scan	Scan_ID Type App_ID (FK) User_ID (FK) Scan_Results Date	Varchar Varchar Varchar Varchar Text DateTime	100 50 100 100
Report	Report_ID Scan_ID (FK) Security Status App Details Permissions Used Intent Analysis	Varchar Varchar Varchar Varchar Varchar Varchar	100 100 50 100 100 100
Notification	Notification_ID User_ID (FK) Message Date Status	Varchar Varchar Varchar DateTime Varchar	100 100 100 100

4.CLASS-BASED MODELING

4.1. CLASS BASED MODELING CONCEPT:

Class-based modelling represents the objects that the system will manipulate, the operations that will be applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

4.2. General Classification:

Candidate classes were then characterized in seven general classes. The seven general characteristics are as follows:

1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

4.3. Noun List:

SL	Noun	General Classification
1	Droid Scanner	2
2	App	1, 2, 7
3	Security	2, 3
4	Management System	2, 7
5	Software	2, 7
6	Solution	2, 7
7	Mobile Device	2, 6
8	Threat	2, 3
9	Standards	2, 7
10	App Development	2, 3
11	Stakeholder	1, 4
12	Groups	5
13	Android Device User	1, 4, 5
14	Device Security	2
15	Android App Developers	1, 4, 5
16	Tools	2, 7
17	Vulnerabilities	2, 3
18	Privacy	2
19	Options	2

SL	Noun	General Classification
20	Full Scan	2, 3
21	Quick Scan	2, 3
22	Command	2, 3
23	Forms	2, 7
24	Security Monitoring	2, 3
25	Tool	2, 7
26	Setup	2, 7
27	User	1, 2, 3, 7
28	Profile	2
29	Scans History	2
30	Blacklist	2
31	Whitelist	2
32	Applist	2, 3
33	Malicious App Detection	2
34	Feature	2, 3
35	APK Files	2
36	APK Extraction	2, 7
37	Custom App Management	2, 7
38	Checks	2, 3
39	Risky Apps	2
40	Scan	2, 3, 7

SL	Noun	General Classification
41	Outcome	2, 7
42	User Database	2, 7
43	Reports	2, 3, 7
44	Scan Summaries	2
45	Data Privacy	2
46	Compliance	2
47	Safety	2, 3
48	Launch	2
49	Dashboard	1, 2, 7
50	Scan Reports	2, 7
51	Permissions	6, 7
52	Location	6, 7
53	Camera	2, 7
54	Permission Viewer	2
55	Request	2
56	Privacy	2
57	Report	2, 7
58	App Details	2, 3
59	Intents	2
60	Feature Analysis	2
61	API Calls	2

SL	Noun	General Classification
62	Status	2
63	Industry Standards	2
64	Issue	2
65	Security Records	2
66	Traceable History	2, 7
67	Improvement	2, 7
68	ML Model	1, 2, 3
69	Notification	2, 3, 7
70	Database	2, 3, 7
71	Integrity	2, 3
72	Applications	3
73	Protection	6, 7

4.4. Verb List:

SL	Verb	SL	Verb
1.	designed	2.	stay
3.	safeguarding	4.	ensure
5.	ensuring	6.	serves
7.	serves	8.	evaluate
9.	prioritize	10.	improve
11.	need	12.	access

13.	ensure	14.	enables
15.	comply	16.	view
17.	provides	18.	let
19.	offering	20.	save
21.	scan	22.	review
23.	ensure	24.	making
25.	operates	26.	shows
27.	initiate	28.	choose
29.	accessing	30.	keep
31.	displays	32.	remove
33.	performs	34.	generates
35.	identify	36.	detailing
37.	analyzing	38.	align
39.	allows	40.	detect
41.	save	42.	stored
43.	examine	44.	ensuring
45.	whitelist	46.	maintain
47.	avoid	48.	release
49.	blacklist	50.	prioritize
51.	store	52.	receive
53.	enables	54.	Empower

55.	helping		

For all lists of Noun general classification is performed. If a noun is an essential entity or fill-up 3 or more classification criteria then it is considered as a potential class. Here potential classes are:

1. User
2. Authentication
3. Android Device User
4. Android App Developer
5. Scan
6. Report
7. ML Model
8. Notification
9. Dashboard
10. App
11. Database

4.5. Selection Criteria:

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Potential Class	Selection Criteria
User	1, 2, 3, 4, 5 (selected)
Android Device User	2, 5, 6 (selected)
Android App Developer	2, 5, 6 (selected)
Scan	1, 2, 3, 4, 5 (selected)
Report	1, 2, 3, 4, 5 (selected)
ML Model	1, 2, 3, 6 (selected)
Notification	1, 2, 3, 4, 5 (selected)
Dashboard	2, 3, 5, 6 (selected)
App	1, 2, 3, 4, 5 (selected)
Database	1, 2, 3, 4, 5 (selected)

4.6. Analysis:

Here the list of Final Potential Classes that fulfills Selection Criteria:

1. Authentication
2. User
3. Android Device User
4. Android App Developer
5. App
6. Scan
7. Report
8. ML Model
9. Notification

10. Dashboard
11. Database

4.7. Class Cards

After identifying our final classes we have generated the following class cards.

Authentication	
Attributes	**Methods**
Email_ID	createAccount()
User_ID	login()
Password	logout()
	forgetPassword()
Responsibilities	**Collaborators**
Authentication of users	Mail System, User

User	
Attributes	**Methods**
User_ID	viewProfile()
Name	updateProfile()
Password	
Role	
Responsibilities	**Collaborators**
Manage user accounts and profiles.	Dashboard

Android Device User	
Attributes	**Methods**
User_ID	viewScanHistory()
Blacklist	addToBlacklist()
Whitelist	addToWhitelist()
	connectDevice()
Responsibilities	**Collaborators**
Manage blacklist and whitelist for apps on connected devices.	App, Dashboard
Connect devices.	Scan

Android App Developer	
Attributes	**Methods**
Developer_ID	viewScanReport()
Uploaded_Apps	uploadAPK()
	viewUploadedApps()
Responsibilities	**Collaborators**
Upload and review APK files for security scanning.	App,Scan
Access reports to ensure compliance with standards.	Dashboard

App	
Attributes	**Methods**
PackageName	checkSecurityStatus()
APK_File	extractAPKDetails()
Version	extractfeatures()
Permissions	updateStatus()
Intents	
Status	
Responsibilities	**Collaborators**
Security check and status update	ML Model
Analyze permissions and intents to meet compliance standards.	ML Model

Scan	
Attributes	**Methods**
Scan_ID	performQuickScan()
Type	performFullScan()
Date	uploadScanResults()
APK_ID	analyzeApp()
Report_ID	
Responsibilities	**Collaborators**
Analyze apps and generate detailed reports.	App,Report

Report	
Attributes	**Methods**
Report_ID	generateScanReport()
Scan_ID	storeScanDetails()
App_ID	
App Details	
Responsibilities	**Collaborators**
Provide Scan Details	Scan
Detail app permissions and intents.	App
Provide Scan Report	Dashboard

ML Model	
Attributes	**Methods**
Model_ID	trainModel()
Classifier	classifyApp()
Trained_Data	improveDatabase()
Results	
Responsibilities	**Collaborators**
Analyze extracted APK data and classify apps as malicious or benign.	Scan, App
Continuously improve the model using new data.	Database

Notification	
Attributes	**Methods**
Notification_ID	sendNotification()
User_ID (FK)	notifyScanResults()
Message	viewNotifications()
Responsibilities	**Collaborators**
Notify users of scan results and detected vulnerabilities.	User, Dashboard

Dashboard	
Attributes	**Methods**
Dashboard_ID	viewUserDashboard()
Scan_History	updateScanHistory()
Connected_Devices	manageConnectedDevices()
Responsibilities	**Collaborators**
Display and update scan history	Report
Show and manage connected devices	User

Database	
Attributes	**Methods**
Report_ID	storeScanReport()
Scan_ID	storeScanDetails()
Permissions	updateDatabase()
Responsibilities	**Collaborators**
Store detailed app permissions and intents.	App

4.7. CRC Diagram:

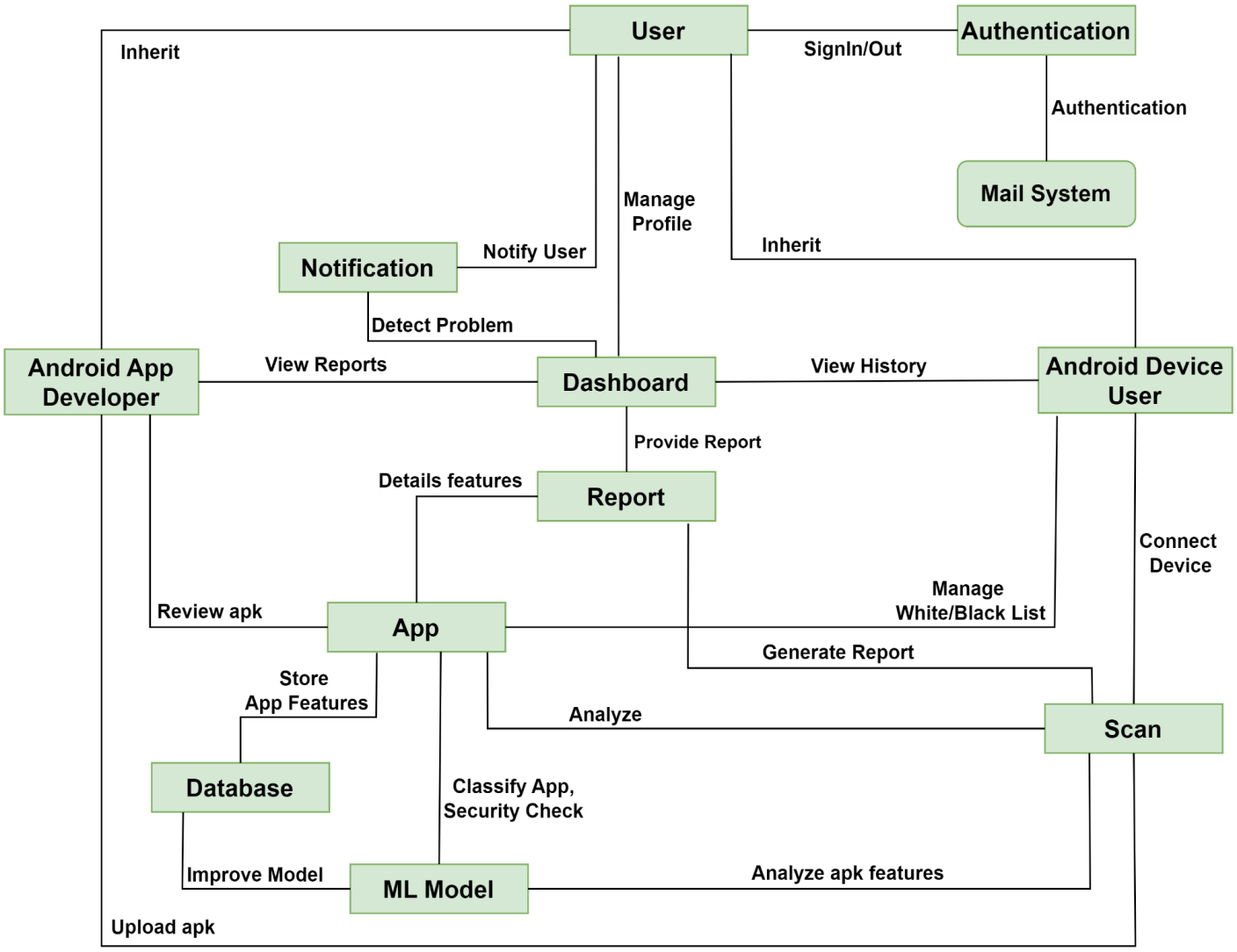

Figure 4

5.Behavioral Modeling

5.1. Definition of Behavioral Modeling :

The behavioral model indicates how software will respond to external events or stimuli. In the context of behavioral modeling, two different characterizations of states must be considered: (1) the state of each class as the system performs its function and (2) the state of the system as observed from the outside as the system performs its function.

State Transition Diagram

One component of a behavioral model is a UML state diagram that represents active states for each class and the events (triggers) that cause changes between these active states.

5.2. Lists of events:

Sl	Event	Initiator	Collaborator
1.	Account Creation	Authentication	User
2.	Profile Update	User	Dashboard
3.	Authentication	Authentication	Mail System
4.	Scan History View	Android Device User	Dashboard
5.	Update Blacklist	Android Device User	
6.	Update Whitelist	Android Device User	
7.	Connect Device	Android Device User	
8.	View Scan Report	User	Dashboard
9.	APK Upload	Android App developers	Scan

Sl	Event	Initiator	Collaborator
10.	Quick Scan	User	Scan
11.	Full Scan	User	Scan
12.	App Analysis	Scan	App
13.	Scan Report Generation	Report	App,ML Model
14.	Store Scan Details	Report	Scan
15.	Train Model	ML Model	
16.	Classify App	ML model	App
17.	Send Notification	Notification	Dashboard
18.	Notify Scan Results	Notification	Dashboard
19.	View Notifications	User	Notification
20.	View User Dashboard	User	Dashboard
21.	Extract APK Details	Scan	App
22.	Status Update	App	ML Model
23.	Store Scan Report	Report	Database

5.3 State Transition Diagram

5.3.1. ID: 1

Name: Authentication

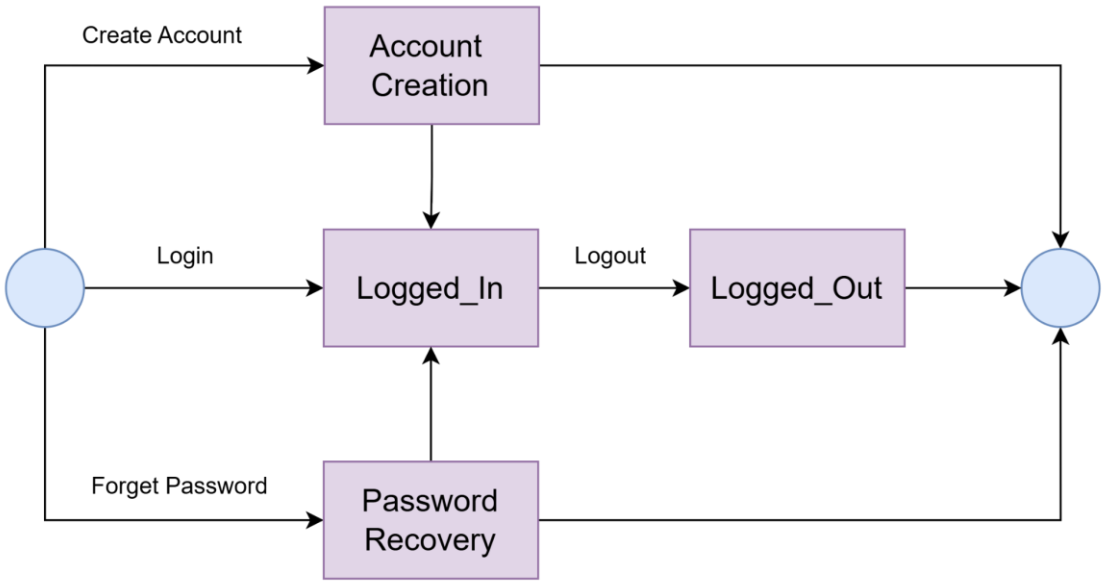

Figure 5.1

5.3.2. ID: 3

Name: Android Device User

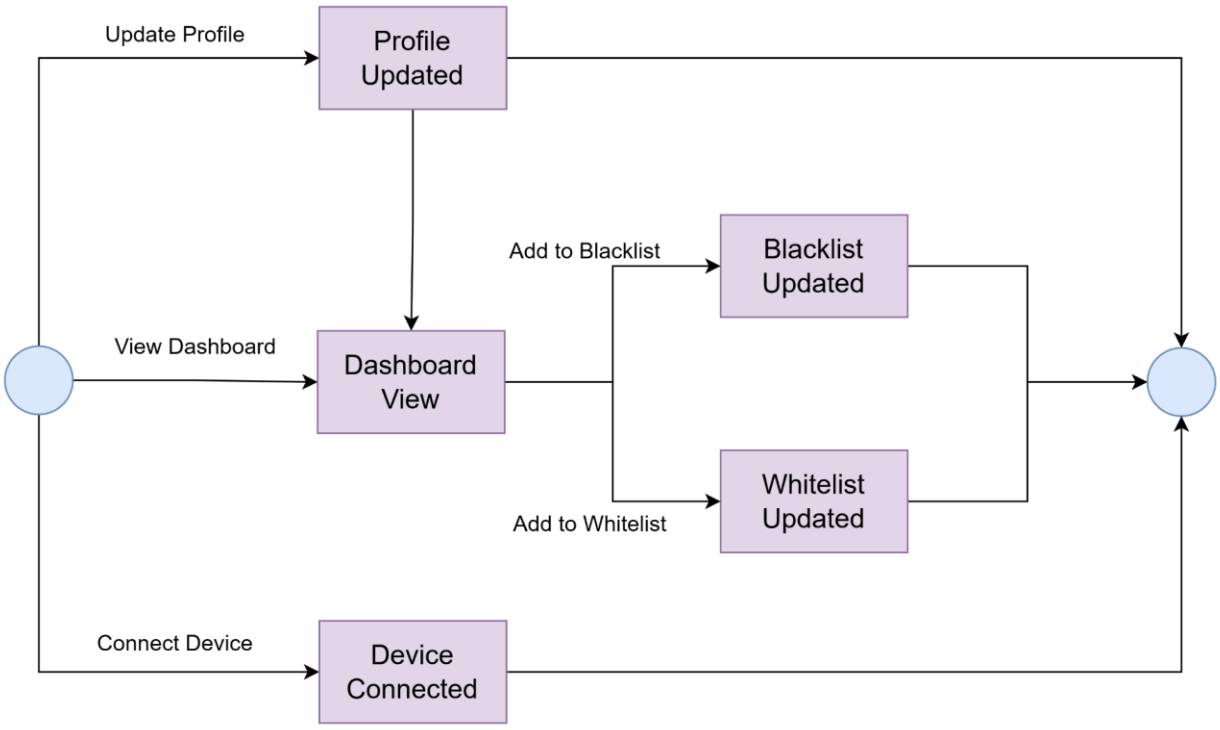

Figure 5.2

5.3.3. ID: 4

Name: Android App Developer

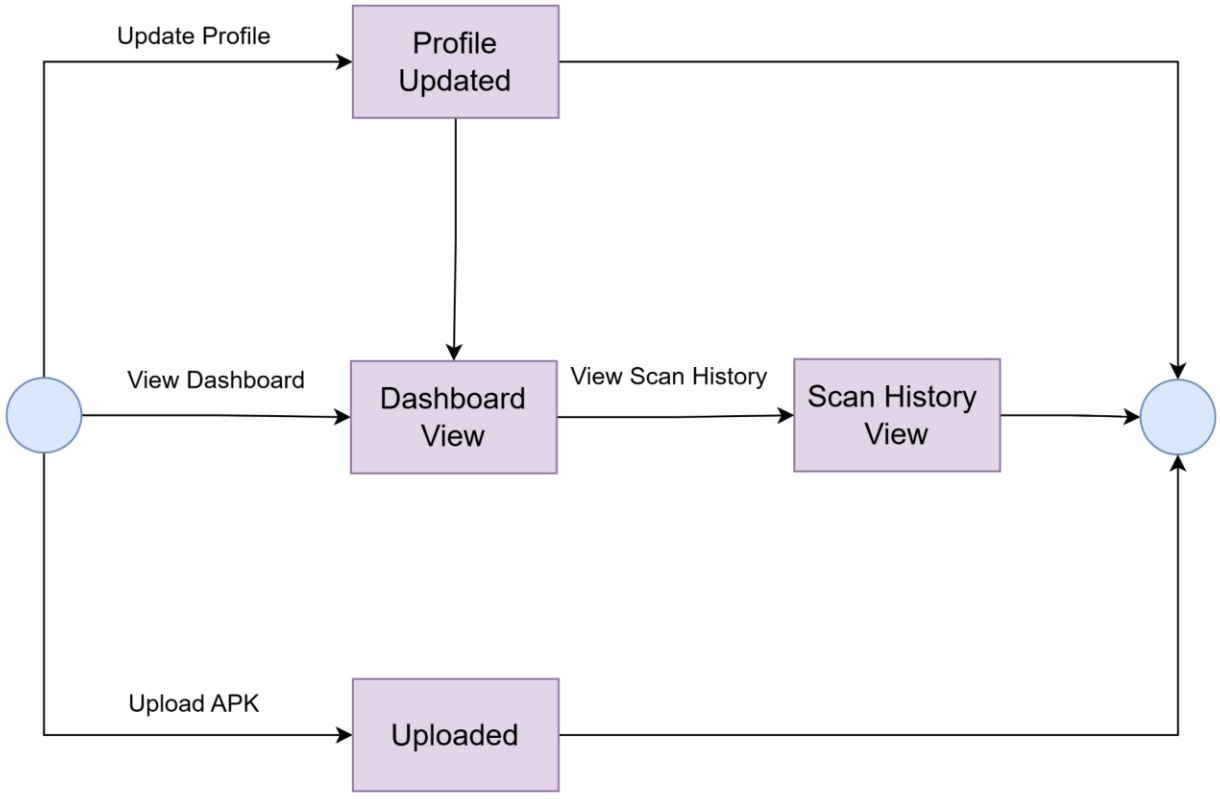

Figure 5.3

5.3.4. ID: 5

Name: App

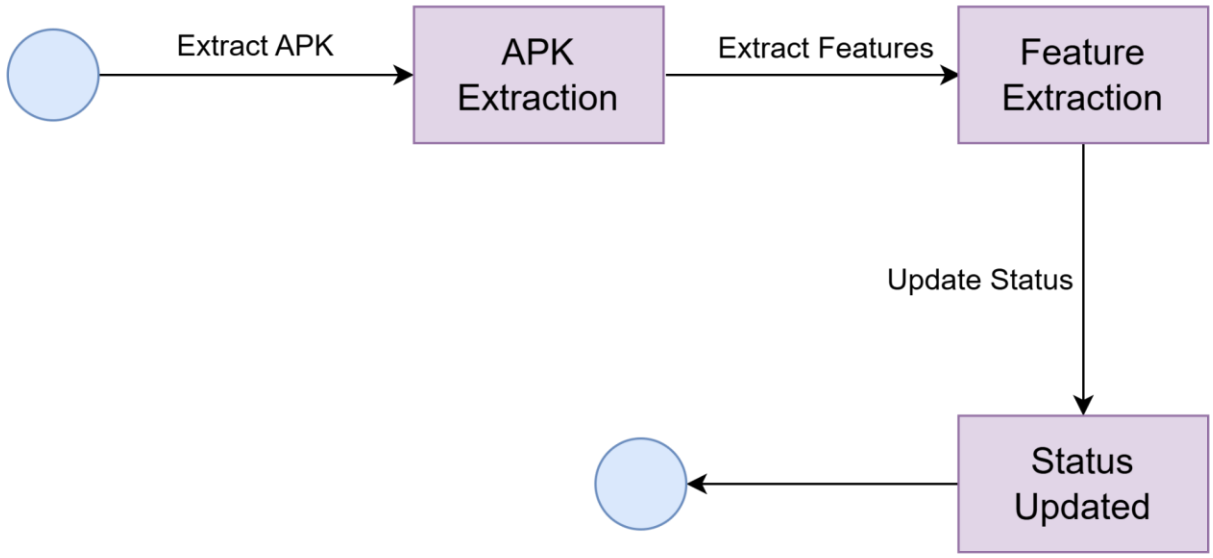

Figure 5.4

5.3.5. ID: 6

Name: Scan

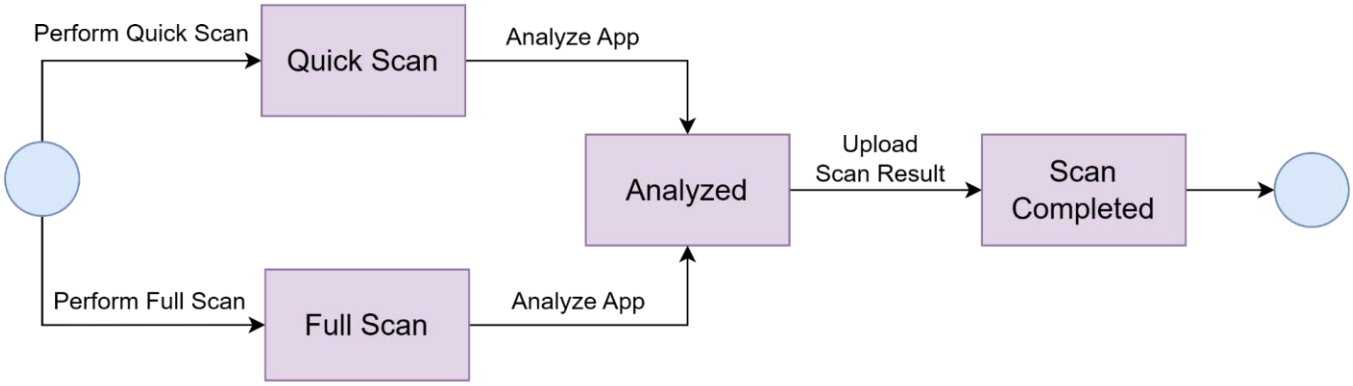

Figure 5.5

5.3.6. ID: 7

Name: Report

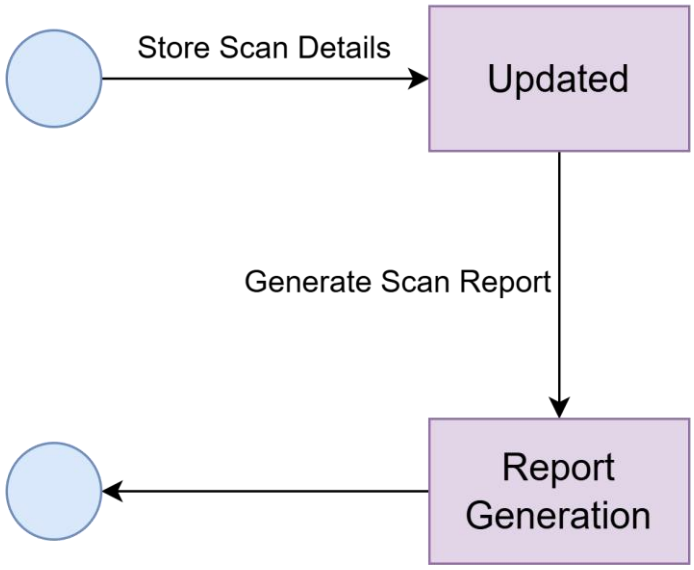

Figure 5.6

5.3.7. ID: 8

Name: ML Model

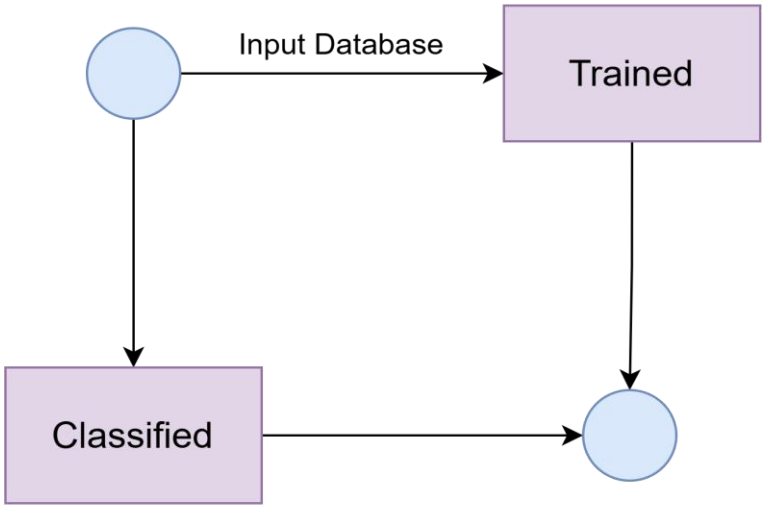

Figure 5.7

5.3.8. ID: 9

Name: Notification

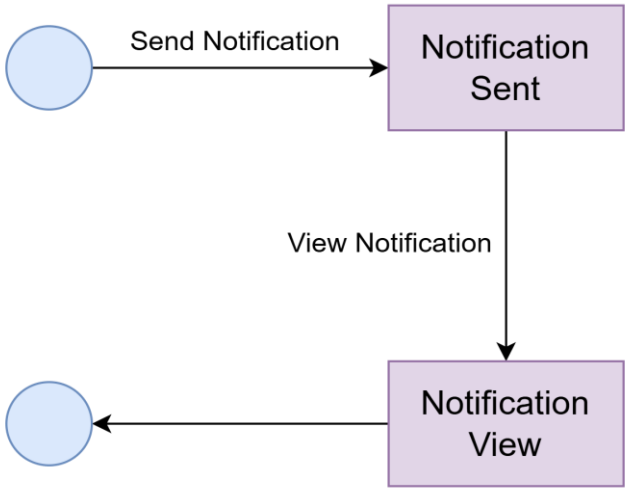

Figure 5.8

5.3.9. ID: 10

Name: Dashboard

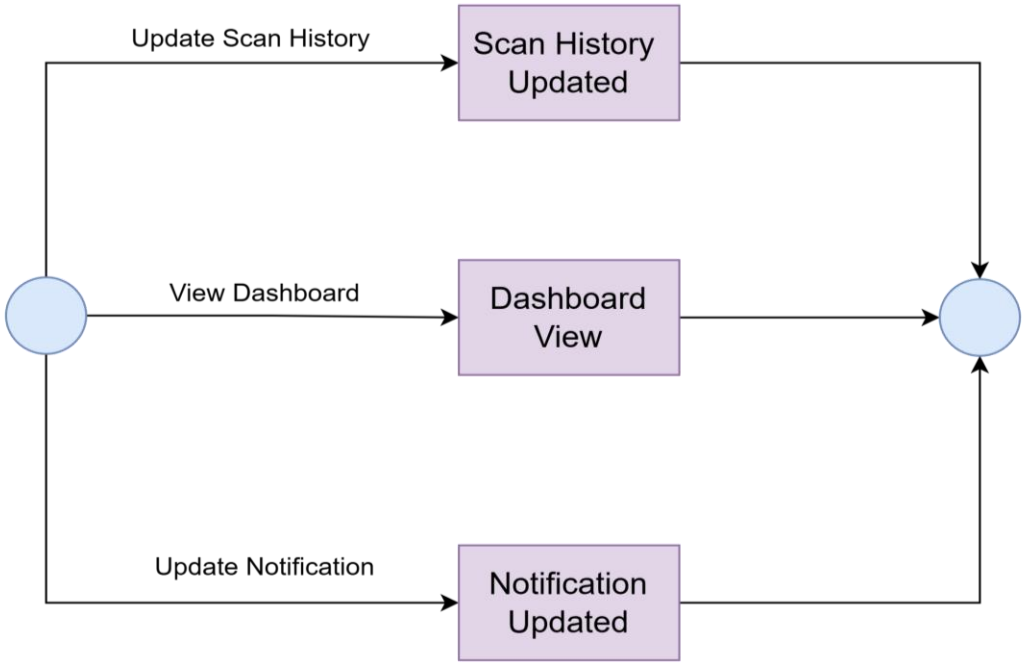

Figure 5.9

5.3.10. ID: 11

Name: Database

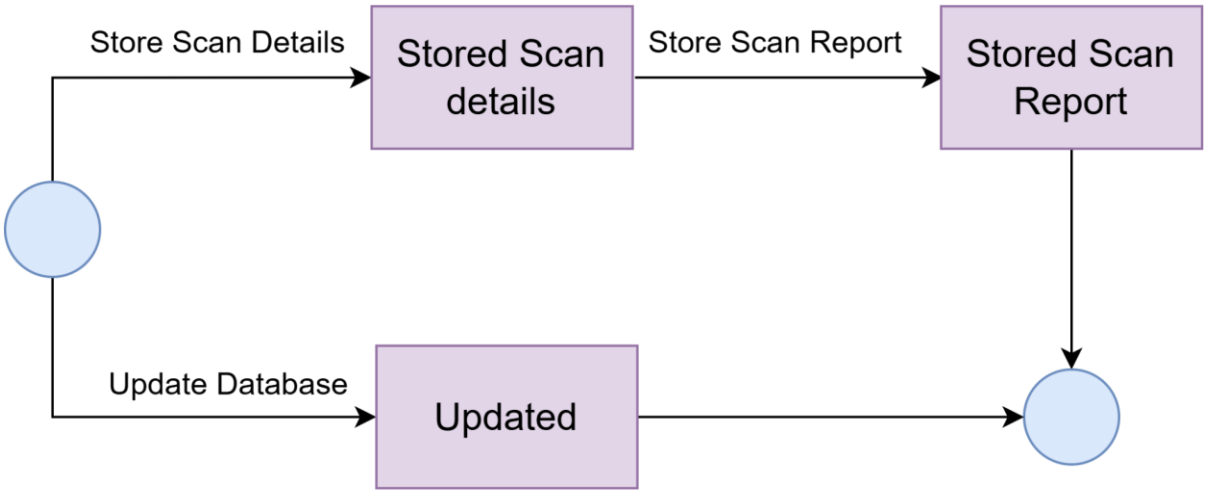

Figure 5.10

6.Sequence Diagram

6.1. Concept of Sequence Diagram :

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. It is a representation of how events cause flow from one object to another as a function of time. In essence, the sequence diagram is a shorthand version of the use case. It represents key classes and the events that cause behavior to flow from class to class.

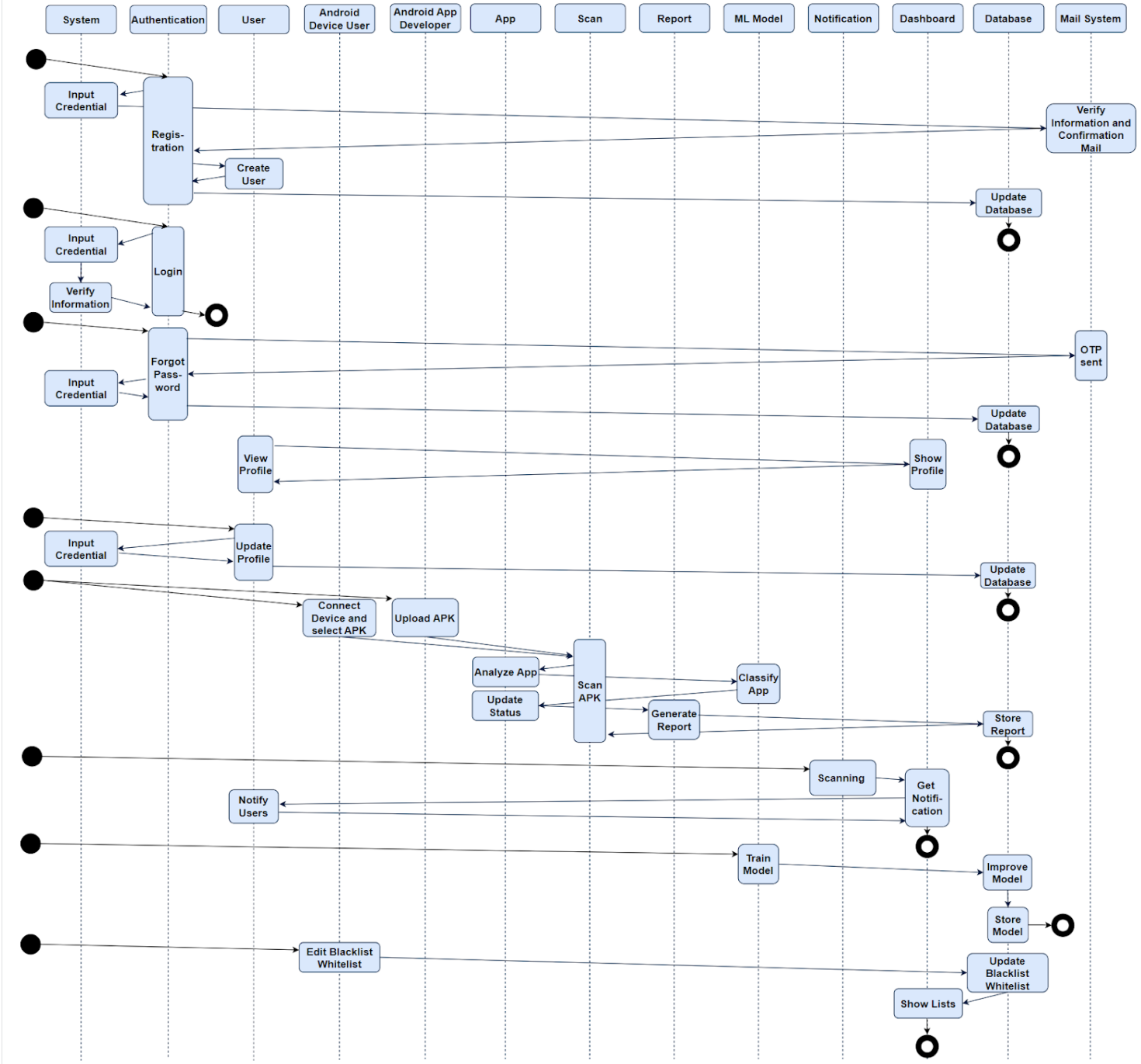
System
Authentication
User
Android Device User
Android App Developer
App
Scan
Report
ML Model
Notification
Dashboard
Database
Mail System
Input Credential
Regis-tration
Verify Information and Confirmation Mail
Create User
Update Database
Input Credential
Login
Verify Information
Forgot Pass-word
OTP sent
Input Credential
Update Database
View Profile
Show Profile
Input Credential
Update Profile
Update Database
Connect Device and select APK
Upload APK
Analyze App
Update Status
Scan APK
Classify App
Generate Report
Store Report
Scanning
Get Notifi-cation
Notify Users
Train Model
Improve Model
Store Model
Edit Blacklist Whitelist
Update Blacklist Whitelist
Show Lists

Figure 6

Conclusion :

We’ve taken the time to really think about who will be using DroidScanner and how it can make their lives easier. This document is here to give everyone involved a clear picture of what the project is about and how it works.

Our main goal is to create software that simplifies app security management for both mobile app developers and everyday users. Whether you’re someone creating apps or just trying to keep your phone safe, DroidScanner is here to make the process simple and stress-free.

We’re designing this tool to be straightforward and easy to use. You won’t need to be a tech wizard to figure it out. Developers will have quick access to app security insights, and users will get clear, actionable information about the apps they use.Above all, we’ve made it a priority to listen to the people who’ll actually use this system. Their feedback has shaped how the software works and ensured it’s practical and helpful for everyone.

With DroidScanner, our aim is simple: to help people manage app security with confidence and ease.

References:

1. *Software_Engineering_A_practitioners_approach_by_roger_s._pressman*
2. *Sommerville-Software-Engineering-10ed*