# 

# 



**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

**Table of Contents**

[User Story : 1](#_Toc184608603)

[QFD (Quality Function Deployment): 3](#_Toc184608604)

[1. Use Case Diagram 4](#_Toc184608605)

[Primary Actor: 4](#_Toc184608606)

[Level 0: 5](#_Toc184608607)

[Level 1: 6](#_Toc184608608)

[Level 1.1: 7](#_Toc184608609)

[Level 1.2 8](#_Toc184608610)

[Level 1.2.1 9](#_Toc184608611)

[Level 1.2.2: 10](#_Toc184608612)

[Level 1.3 11](#_Toc184608613)

[Level 1.4 12](#_Toc184608614)

[2.Swimlane Diagram 13](#_Toc184608615)

[2.1. Definition of Swimlane Diagram 13](#_Toc184608616)

[2.2. Swimlane ID (SID) 1.1 14](#_Toc184608617)

[2.3. Swimlane ID (SID) 1.2 15](#_Toc184608618)

[2.3. Swimlane ID (SID) 1.2.1 16](#_Toc184608619)

[2.4. Swimlane ID (SID) 1.2.2 17](#_Toc184608620)

[2.5. Swimlane ID (SID) 1.3 18](#_Toc184608621)

[2.6. Swimlane ID (SID) 1.4 19](#_Toc184608622)

[3.Data Based Modeling 20](#_Toc184608623)

[3.1. Definition of Database Diagram 20](#_Toc184608624)

[3.2. Data Objects Identification: 20](#_Toc184608625)

[3.3. Final Data Objects: 24](#_Toc184608626)

[3.4. Relationship between Objects: 24](#_Toc184608627)

[3.5. Entity Relation ( ER ) Diagram: 25](#_Toc184608628)

[3.6. Schema Diagram: 26](#_Toc184608629)

[4.CLASS-BASED MODELING 27](#_Toc184608630)

[4.1. CLASS BASED MODELING CONCEPT: 27](#_Toc184608631)

[4.2. General Classification: 27](#_Toc184608632)

[4.3. Noun List: 28](#_Toc184608633)

[4.4. Verb List: 31](#_Toc184608634)

[4.5. Selection Criteria: 33](#_Toc184608635)

[4.6. Analysis: 34](#_Toc184608636)

[4.7. Class Cards 35](#_Toc184608637)

[4.7. CRC Diagram: 40](#_Toc184608638)

[5.Behavioral Modeling 41](#_Toc184608639)

[5.1. Definition of Behavioral Modeling : 41](#_Toc184608640)

[5.2. Lists of events: 41](#_Toc184608641)

[5.3 State Transition Diagram 43](#_Toc184608642)

[5.3.1. ID: 1: Authentication 43](#_Toc184608643)

[5.3.2. ID: 2: Android Device User 44](#_Toc184608644)

[5.3.3. ID: 3: Android App Developer 45](#_Toc184608645)

[5.3.4. ID: 4: App 46](#_Toc184608646)

[5.3.5. ID: 5: Scan 46](#_Toc184608647)

[5.3.6. ID: 6: Report 47](#_Toc184608648)

[5.3.7. ID: 7: ML Model 47](#_Toc184608649)

[5.3.8. ID: 8: Notification 48](#_Toc184608650)

[5.3.9. ID: 9: Dashboard 48](#_Toc184608651)

[5.3.10. ID: 10: Database 49](#_Toc184608652)

[6.Sequence Diagram 49](#_Toc184608653)

[6.1. Concept of Sequence Diagram : 49](#_Toc184608654)

**List Of Figures**

[Figure 0: ASMS](#_Level_0:) 5

[Figure 1: ASMS(Detailed)](#_Level_1:) 6

[Figure 1.1: Authentication](#_Level_1.1:) 7

[Figure 1. 2: Malicious App Detection](#_Level_1.2) 8

[Figure 1.2.1: Scan](#_Level_1.2.1) 9

[Figure 1.2.2: APK Analysis](#_Level_1.2.2:) 10

[Figure 1.3: Custom App Management](#_Level_1.3) 11

[Figure 1.4: Report Generation](#_Level_1.4) 12

[Figure: SID 1.1](#_2.2._Swimlane_ID) 14

[Figure: SID 1.2](#_2.3._Swimlane_ID) 15

[Figure: SID 1.2.1](#_2.3._Swimlane_ID_1) 16

[Figure: SID 1.2.2](#_2.4._Swimlane_ID) 17

[Figure: SID 1.3](#_2.5._Swimlane_ID) 18

[Figure: SID 1.4](#_2.6._Swimlane_ID) 19

[Figure 3: Entity Relation ( ER ) Diagram](#_3.5._Entity_Relation) 25

[Figure 4: CRC Diagram](#_4.7._CRC_Diagram:) 40

[Figure 5.1: Authentication](#_5.3.1._ID:_1) 43

[Figure 5.2: Android Device User](#_5.3.2._ID:_2) 44

[Figure 5.3: Android App Developer](#_5.3.3._ID:_3) 45

[Figure 5.4: App](#_5.3.4._ID:_4) 46

[Figure 5.5: Scan](#_5.3.5._ID:_5) 46

[Figure 5.6: Report](#_5.3.6._ID:_6) 47

[Figure 5.7: ML Model](#_5.3.7._ID:_7) 47

[Figure 5.8: Notification](#_5.3.8._ID:_8) 48

[Figure 5.9: Dashboard](#_5.3.9._ID:_9) 48

[Figure 5.10: Database](#_5.3.10._ID:_10) 49

[Figure 6: Sequence Diagram](#_6.1._Concept_of) 50

**Software Requirement Specification report of**

**“Droid Scanner: App Security Management System”**

# Introduction:

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.

It is a comprehensive tool designed to analyze Android applications (APK files) for security vulnerabilities, suspicious permissions, and potential malicious intents. It integrates role-based access control (Personal, Business, Admin) with secure OTP-based authentication and machine learning (ML) to classify threats. The system ensures secure user management, real-time APK scanning, and detailed reporting, making it suitable for both individual users and organizations.

### \*\*2. Project Overview\*\*

#### \*\*2.1 Key Features\*\*

1. \*\*Secure Authentication\*\*:

   - Email verification via 6-digit OTPs sent using SMTP (Gmail).

   - JWT tokens for session management with 5-minute expiration.

2. \*\*Role-Based Access Control\*\*:

   - \*\*Personal Users\*\*: Basic APK scanning and threat reports.

   - \*\*Business Users\*\*: Restricted registration to organizational domains (e.g., \*@company.com).

   - \*\*Admins\*\*: Access to ML model training and system analytics.

3. \*\*APK Analysis\*\*:

   - Extract permissions and intents from `AndroidManifest.xml`.

   - Classify apps as \*\*Benign\*\* or \*\*Malicious\*\* using a KNN model.

4. \*\*User-Friendly GUI\*\*:

   - PyQt5-based interface with dynamic workflows for login, registration, and scanning.

   - Real-time scan results with color-coded risk levels.

#### \*\*2.2 Technical Stack\*\*

| \*\*Component\*\*       | \*\*Technology/Tool\*\*              |

|----------------------|-----------------------------------|

| Frontend             | PyQt5, Qt Designer               |

| Backend              | Python 3.10                      |

| Database             | SQLite (User data), App Data     |

| Security             | JWT, SMTP (Gmail)                |

| Machine Learning     | Scikit-learn, Pandas, Joblib     |

| APK Parsing          | ADB, Apktool                     |

---

### \*\*3. System Architecture\*\*

#### \*\*3.1 Authentication Workflow\*\*

1. \*\*Registration/Login\*\*:

   - Users provide email, username, and password.

   - Business users must use allowed organizational domains (e.g., \*@company.com).

2. \*\*OTP Verification\*\*:

   - A 6-digit OTP is generated and sent via email.

   - JWT token is created with OTP and email data.

3. \*\*Session Management\*\*:

   - Token is stored in the database and validated for subsequent requests.

\*\*Code Snippet: OTP Generation & JWT Creation\*\*

```python

# Authentication.py

def generate\_otp(self):

    return str(random.randint(100000, 999999))  # 6-digit OTP

def create\_jwt(self, email, otp):

    payload = {

        "email": email,

        "otp": otp,

        "exp": datetime.utcnow() + timedelta(minutes=5)

    }

    return jwt.encode(payload, SECRET\_KEY, algorithm="HS256")

```

#### \*\*3.2 Database Design\*\*

- \*\*Users Table\*\*:

  ```sql

  CREATE TABLE users (

      id INTEGER PRIMARY KEY AUTOINCREMENT,

      username TEXT UNIQUE NOT NULL,

      password TEXT NOT NULL,

      email TEXT UNIQUE,

      verified INTEGER DEFAULT 0,  -- 0=Unverified, 1=Verified

      verification\_token TEXT,

      user\_type TEXT  -- "personal", "business", "admin"

  )

  ```

- \*\*App Data\*\*: Stores APK metadata, permissions, and scan results.

#### \*\*3.3 ML Model Integration\*\*

- \*\*Training Data\*\*: `processed\_output.csv` (pre-processed dataset of app permissions/intents).

- \*\*Algorithm\*\*: K-Nearest Neighbors (KNN) with `n\_neighbors=1`.

- \*\*Workflow\*\*:

  1. Extract features (permissions/intents) from APK.

  2. Convert features into a binary vector (1=present, 0=absent).

  3. Predict using the trained model.

\*\*Code Snippet: ML Model Training\*\*

```python

# MLmodel.py

def train\_model(self):

    df = pd.read\_csv(self.csv\_file)

    X = df.drop(columns=['prediction'])  # Features

    Y = df['prediction']                # Labels (Benign/Malicious)

    X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2)

    self.classifier = KNeighborsClassifier(n\_neighbors=1)

    self.classifier.fit(X\_train, Y\_train)

    joblib.dump(self.classifier, 'AppClassifier1.joblib')  # Save model

```

---

### \*\*4. Implementation\*\*

#### \*\*4.1 Authentication Module\*\*

- \*\*OTP Handling\*\*:

  - Generated using Python’s `random` module.

  - Sent via Gmail SMTP using `smtplib`.

- \*\*Token Validation\*\*:

  - Tokens are decoded with `jwt.decode()` and checked for expiration.

\*\*Code Snippet: Email Sending\*\*

```python

# Authentication.py

def send\_email(self, recipient, otp):

    server = smtplib.SMTP("smtp.gmail.com", 587)

    server.starttls()

    server.login(EMAIL, EMAIL\_PASSWORD)

    server.sendmail(EMAIL, recipient, msg.as\_string())

```

#### \*\*4.2 Role-Based GUI (PyQt5)\*\*

- \*\*AuthWindow.py\*\*:

  - Toggles between login/registration modes.

  - Validates business user emails against allowed domains.

  ```python

  # AuthWindow.py

  ALLOWED\_ORG\_DOMAINS = ['company.com', 'enterprise.org']

  if self.user\_type == "business":

      if not email.endswith(tuple(ALLOWED\_ORG\_DOMAINS)):

          QMessageBox.warning("Domain Error: Use organizational email.")

  ```

- \*\*MainWindow.py\*\*:

  - Drag-and-drop APK upload.

  - Displays scan results with risk summaries.

#### \*\*4.3 APK Scanning Engine\*\*

1. \*\*Device Connection\*\*:

   - Uses ADB to detect connected Android devices.

2. \*\*APK Extraction\*\*:

   - Pulls `base.apk` from the device.

3. \*\*Permission/Intent Extraction\*\*:

   - Uses `apktool` to decompile APK and parse `AndroidManifest.xml`.

\*\*Code Snippet: APK Extraction\*\*

```python

# scan.py

def extract\_apk(self):

    subprocess.run(["adb", "pull", base\_apk\_path, local\_apk\_path])

    permissions = parsing.extract\_permissions(manifest\_path)

    intents = parsing.extract\_intents(manifest\_path)

```

---

### \*\*5. User Manual\*\*

#### \*\*5.1 Personal User Guide\*\*

1. \*\*Registration\*\*:

   - Enter email, username, password.

   - Verify via OTP sent to your email.

2. \*\*Login\*\*:

   - Use credentials to access the dashboard.

3. \*\*Scan APK\*\*:

   - Upload APK or connect an Android device.

   - View results with highlighted risks.

#### \*\*5.2 Business/Admin Guide\*\*

1. \*\*Registration\*\*:

   - Use a company email (e.g., \*@company.com).

2. \*\*Advanced Features\*\*:

   - \*\*Train ML Model\*\*: Click "Train Model" to update the classifier.

   - \*\*View Reports\*\*: Detailed permissions/intents list with risk levels.

---

### \*\*6. Challenges Faced\*\*

1. \*\*GUI-Backend Integration\*\*:

   - Synchronizing PyQt5 signals/slots with Python backend logic.

2. \*\*Token Expiry Handling\*\*:

   - Managing JWT expiration and OTP resend functionality.

3. \*\*APK Parsing\*\*:

   - Decompiling large APKs and handling corrupted files.

---

### \*\*7. Limitations\*\*

1. \*\*Email Dependency\*\*: Relies solely on Gmail for OTP dispatch.

2. \*\*Model Accuracy\*\*: Limited by training data quality and feature selection.

3. \*\*No Real-Time Monitoring\*\*: Cannot detect threats during runtime on devices.

---

### \*\*8. Conclusion & Future Work\*\*

The system successfully integrates secure authentication, role-based access, and ML-driven APK analysis. Future enhancements include:

- \*\*Multi-Email Support\*\*: Add Outlook, Yahoo, etc., for OTPs.

- \*\*Enhanced ML Model\*\*: Use deep learning for better accuracy.

- \*\*Real-Time Protection\*\*: Monitor apps during installation/execution.

---

### \*\*9. Screenshots & Code\*\*

\*\*Figure 1: Authentication Workflow\*\*

```

User → Enter Credentials → OTP Sent → Token Generated → Access Granted

```

\*\*Figure 2: APK Scanning Process\*\*

```

Device → ADB → Extract APK → Parse Permissions → ML Classification → Report

```

\*\*Submitted By\*\*:

- \*\*Your Name\*\*

- \*\*Roll Number\*\*

- \*\*Supervised by: Dr. XYZ\*\*

- \*\*Date: [Insert Date]\*\*

---

**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*