**Project Report on**

**PM-JAY Live Audit PDF Generation App with Firebase Authentication**

**Abstract**

This project report presents the design and development of the **PM-JAY Live Audit App**, a mobile application built with Flutter to digitize the audit process under the Pradhan Mantri Jan Arogya Yojana (PM-JAY). The app integrates **Firebase Authentication** to ensure secure auditor access and connects to the **SurveyCTO API** for fetching real-time audit data. Retrieved data is parsed, validated, and converted into **structured PDF reports** following a standardized template, which improves consistency and readability. The system addresses major challenges of manual audits—such as delays, errors, and reliance on physical records—by enabling **digital record-keeping, faster reporting, and simplified data sharing**. Additional features like local storage and report sharing further streamline the auditor’s workflow. The report discusses the **system requirements, architecture, design, implementation, and testing strategies**, highlighting how the app ensures reliability and functional correctness. Results confirm its effectiveness in improving audit efficiency, while recommendations for **future enhancements** include advanced analytics, offline functionality, and improved visualization.

**Table of Contents**

1. Introduction
2. Literature Review
3. System Analysis
4. System Design
5. System Implementation
6. System Testing
7. Results & Discussion
8. Conclusion & Future Work
9. References

**Chapter 1: Introduction**

**1.1 Background**

Pradhan Mantri Jan Arogya Yojana (PM-JAY) is India’s flagship government health insurance scheme aimed at providing financial protection for secondary and tertiary care hospitalization. To ensure transparency, accountability, and proper implementation, **continuous monitoring and auditing** of enrolled hospitals and claims is essential. Traditionally, these live audits have been carried out manually using paper-based forms or spreadsheets. While functional, such approaches are **time-consuming, prone to errors, difficult to standardize, and cumbersome to store and share**.

With the growing adoption of **digital data collection platforms such as SurveyCTO**, much of the raw audit data is now collected electronically. However, auditors still face challenges in **retrieving this data, validating it securely, and preparing standardized reports** in a timely manner. This gap highlights the need for a dedicated **mobile application** that can directly fetch audit data from SurveyCTO, process it, and generate structured outputs suitable for official reporting.

To ensure that access to sensitive health-audit data remains secure, the application leverages **Google Firebase Authentication** for user management. Unlike storing credentials locally, Firebase provides a **scalable, cloud-based authentication solution** that protects user identities, prevents unauthorized access, and simplifies user account management. This is particularly valuable in the healthcare audit domain, where **confidentiality, integrity, and availability of data** are critical.

**1.2 Problem Statement**

The current process of conducting live audits under PM-JAY involves multiple **manual and fragmented steps**, such as downloading raw data from SurveyCTO, validating it manually, formatting it in Excel/Word, and preparing final reports. These steps are not only **time-consuming and repetitive** but also **susceptible to human error**. Moreover, report formats often vary between auditors, leading to inconsistencies in documentation.

There is a clear need for a **digital solution** that automates key steps of the workflow—namely, **secure authentication, real-time data fetching, structured processing, and standardized PDF report generation**—to improve audit efficiency, consistency, and accuracy.

**1.3 Objectives**

The objectives of this project are as follows:

* To design and implement a mobile application for PM-JAY live audits using the Flutter framework.
* To integrate SurveyCTO API for fetching real-time audit data directly into the app.
* To parse and map survey data into structured formats suitable for reporting.
* To generate audit-ready PDF reports based on a predefined, standardized template.
* To enable auditors to preview, save, and share reports directly from the mobile device.
* To ensure secure user authentication using Firebase, protecting sensitive health-audit data.

**1.4 Scope**

The PM-JAY Live Audit App is designed primarily for **auditors and administrators** engaged in monitoring hospitals and claim records under the PM-JAY scheme. The app will be deployed on the **Android platform**, given its wide adoption among field-level users. Key features of the system include:

* **Secure login and credential management** via Firebase Authentication.
* **Real-time audit data retrieval** from SurveyCTO.
* **Automated PDF report generation** with standardized templates.
* **Offline storage support** for generated reports, ensuring accessibility even without internet connectivity.
* **Report sharing functionality** for seamless communication between auditors, administrators, and higher authorities.

The current scope is limited to **mobile-based reporting** and does not include advanced analytics or predictive fraud detection, which are identified as potential **future extensions** of the system.

**Chapter 2: Literature Review**

* **Manual audit methods**: Slow, error-prone, requires physical storage [1].
* **Survey tools**: Tools like ODK, SurveyCTO, and KoboCollect are widely used for digital data collection but lack direct PDF generation features [3]–[5].
* **PDF generation solutions**: Existing apps allow PDF creation but not tailored to PM-JAY’s auditing process [6], [7].
* **Gap**: No dedicated solution exists for **fetching SurveyCTO audit data and producing PM-JAY compliant PDF reports [1], [2], [9]**.
* **Firebase Authentication**: widely used in mobile applications for secure login and credential management [10]. Cloud-backed user authentication is recommended for e-governance and healthcare applications due to its scalability and enhanced security [11].

**Chapter 3: System Analysis**

**3.1 Requirement Analysis**

**Functional Requirements**

* Fetch audit data via SurveyCTO API.
* The system shall allow secure login via Firebase Authentication before accessing SurveyCTO APIs.
* Parse data into models.
* Generate structured PDF reports.
* Preview, save, and share reports.
* Secure credential storage.

**Non-Functional Requirements**

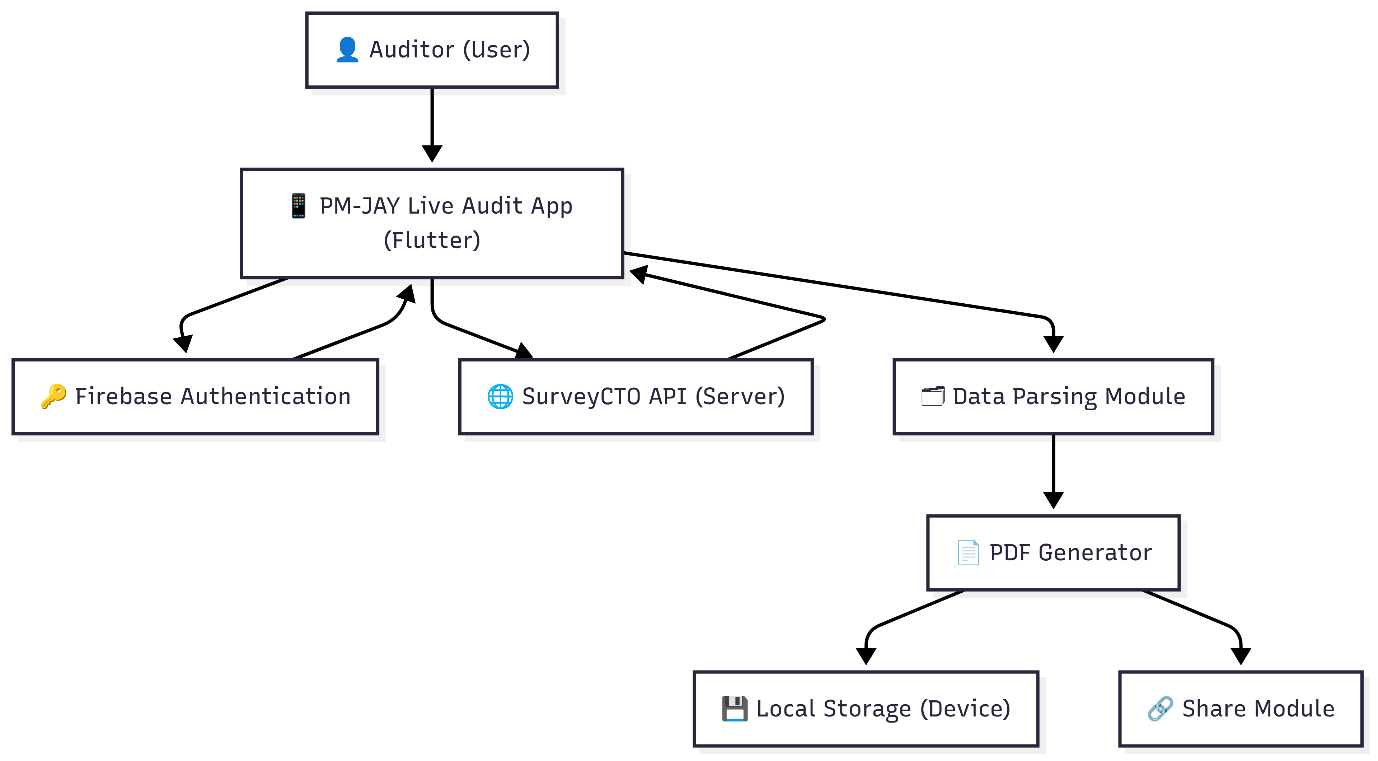
* Usability: Simple, intuitive UI for auditors.
* Security and Management: User credentials must be securely stored and managed through Firebase, ensuring data protection and compliance.
* Performance: Generate PDF in under 20 seconds.
* Security: Store API credentials securely.
* Portability: Cross-platform (Android, iOS).

**3.2 Feasibility Study**

* **Technical Feasibility**: Flutter and SurveyCTO API are stable technologies.
* **Economic Feasibility**: Cost-effective as it is open-source + minimal infrastructure.
* **Operational Feasibility**: Easy to use for auditors in the field.

**Chapter 4: System Design**

**4.1 Architecture**



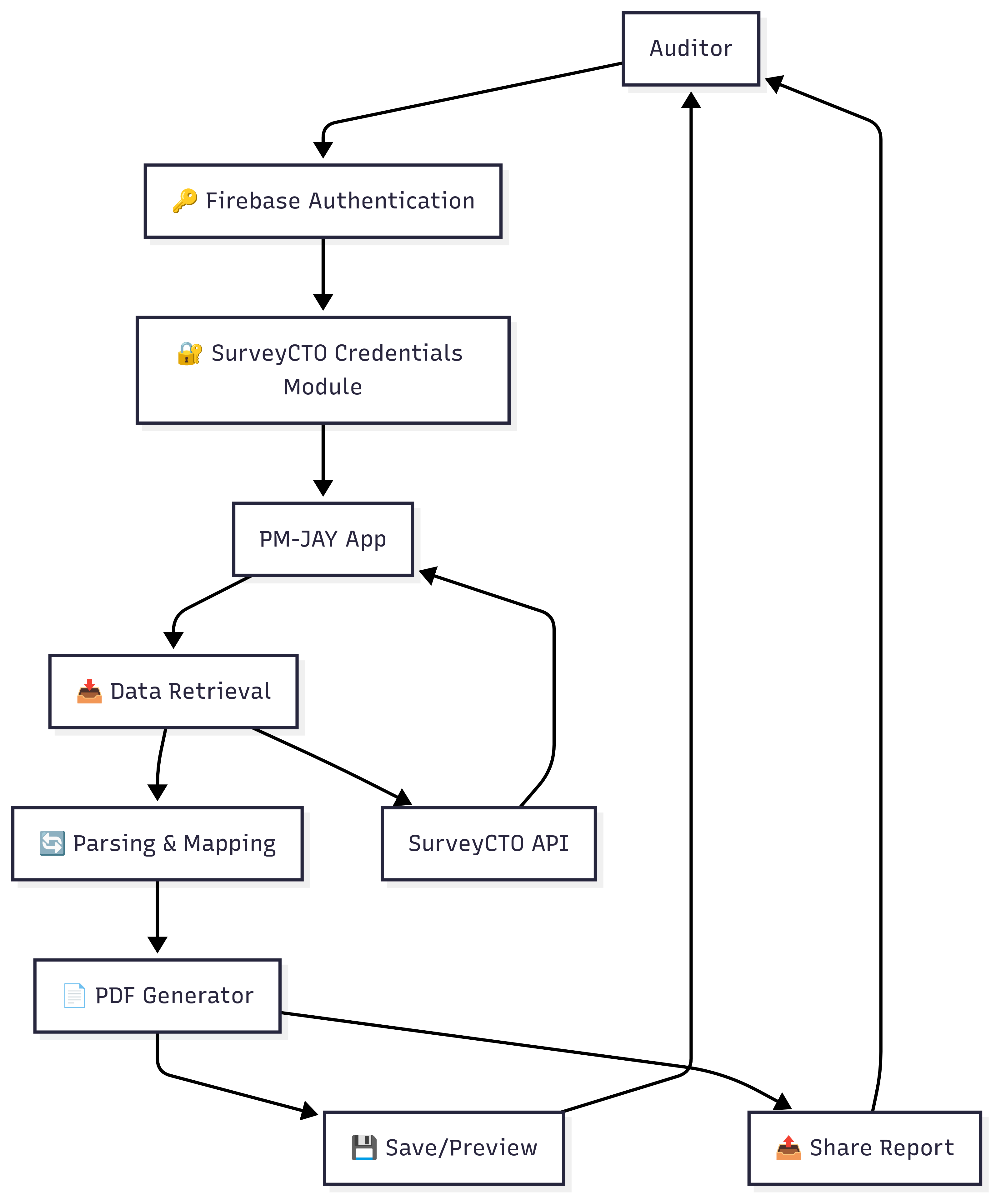
**Flow:**  
SurveyCTO API → API Service → Data Parsing → PDF Generator → Storage/Sharing → User

**4.2 Data Flow Diagram (DFD)**

* **Level 0**: User → App → SurveyCTO API → PDF Output

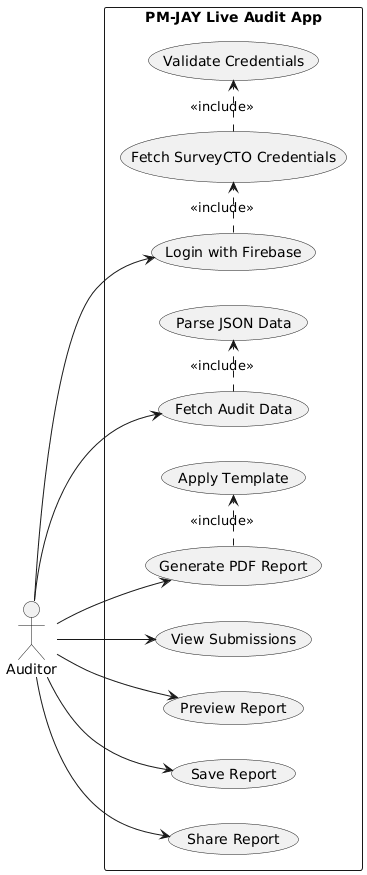


* **Level 1**: Fetch Forms → Fetch Submissions → Parse → Generate PDF

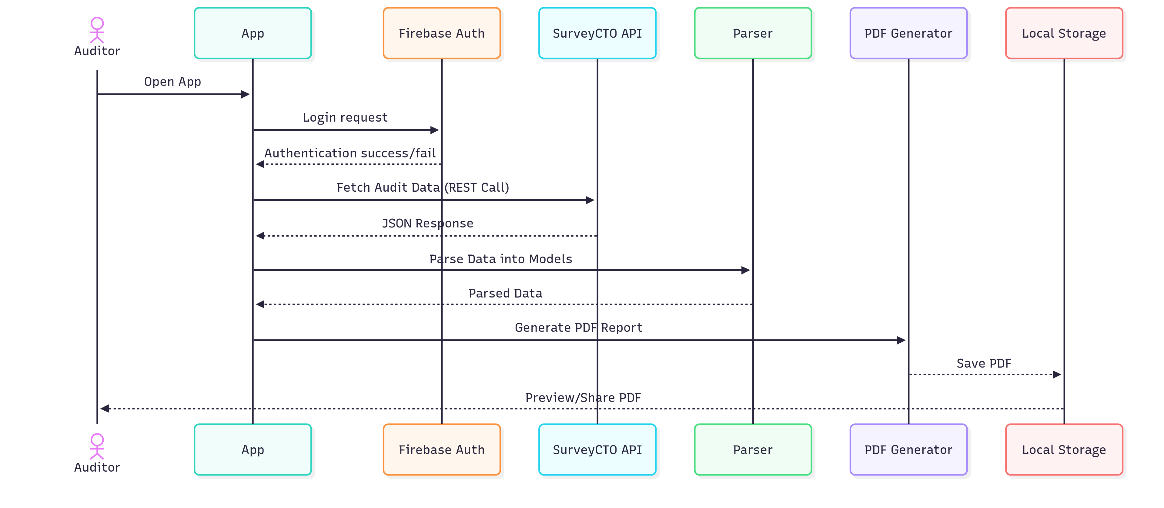


**4.3 UML Diagrams**

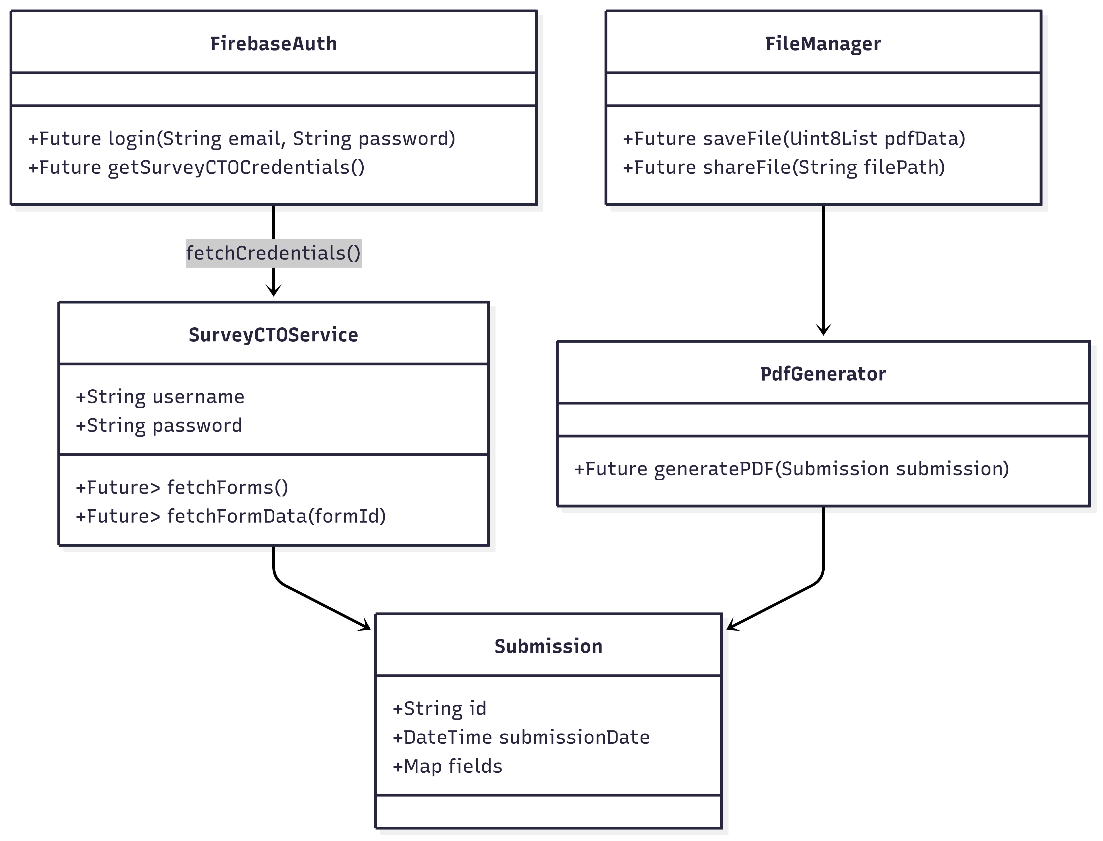
* **Use Case**: Generate Audit PDF, Fetch Data, Share PDF.



* **Sequence Diagram**: Submission selection → PDF generation → Save/Share.



* **Class Diagram**: SurveyCTOService, Submission, PdfGenerator.



**4.4 User Interface Design**

|  |  |  |
| --- | --- | --- |
| Splash screen | Login screen | Hospital and case id wise search screen |
| Hospital and date wise search screen |

**Chapter 5: System Implementation**

**5.1 Technology Stack**

* **Frontend**: Flutter (Dart)
* **Backend**: SurveyCTO API (REST), Google Firebase
* **Libraries**: http, pdf, printing, path\_provider, flutter\_secure\_storage
* **Platforms**: Android

**5.2 Modules**

1. **Authentication Module**
   * Handles login & credential storage.
2. **Data Retrieval Module**
   * Fetches form submissions from SurveyCTO.
3. **Data Parsing Module**
   * Converts JSON into Dart models.
4. **PDF Generator Module**
   * Generates PDFs from parsed data using predefined template.
5. **File Handling Module**
   * Saves, previews, and shares PDFs.

**5.3 Code Snippets**

**5.3.1 Authentication of login credentials to Google Firebase while login**

*try* {

*await* FirebaseAuth.instance.signInWithEmailAndPassword(

*email*: \_userController.text.trim(),

*password*: \_passController.text.trim(),

      );

*final* user = FirebaseAuth.instance.currentUser;

*if* (user == *null*) {

*throw* Exception("Not logged in");

      }

*// Navigate if success*

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(*builder*: (*\_*) => *const* HomeScreen()),

      );

    } *on* FirebaseAuthException *catch* (e) {

      setState(() {

        \_error = e.message ?? 'Login failed';

      });

    } *finally* {

      setState(() {

        \_loading = false;

      });

    }

**5.3.2 SurveyCTO credentials loading from Google Firebase**

static Future<void> loadCredentials() async {

if (\_server != null) return; // already loaded

final doc = await FirebaseFirestore.instance

.collection('config')

.doc('surveycto')

.get();

if (doc.exists) {

final data = doc.data()!;

\_server = data['server'];

\_formId = data['formId'];

\_username = data['username'];

\_password = data['password'];

} else {

throw Exception("SurveyCTO config not found in Firestore");

}

}

**5.3.3 Finding records using Hospital ID and Case ID:**

*static* Future<List<Map<String, dynamic>>> fetchFormData() *async* {

*await* loadCredentials();

*final* url = Uri.https(\_server!, '/api/v1/forms/data/wide/json/$*\_formId*');

*final* response = *await* http.get(

      url,

*headers*: {

        'Authorization':

            'Basic ' + base64Encode(utf8.encode('$*\_username*:$*\_password*')),

      },

    );

*if* (response.statusCode == 200) {

*final* List<dynamic> data = json.decode(response.body);

*return* data.cast<Map<String, dynamic>>();

    } *else* {

*throw* Exception('Failed to fetch data: ${*response*.*statusCode*}');

    }

  }

*static* Future<Map<String, dynamic>?> findRecord(

    String *hospitalId*,

    String *caseNo*,

  ) *async* {

*final* data = *await* fetchFormData();

*for* (*final* entry *in* data) {

*if* ((entry['hospital\_id'] ?? '').toString().trim() == *hospitalId*.trim() &&

          (entry['case\_no'] ?? '').toString().trim() == *caseNo*.trim()) {

*return* \_normalizeRecord(entry);

      }

    }

*return* *null*;

  }

**Chapter 6: Testing**

**6.1 Overview**

This chapter presents the test cases for the PMJAY Live Audit App. The test cases are structured to ensure:

* Each module functions correctly (Unit Tests)
* Modules integrate seamlessly (Integration Tests)
* The app handles errors and edge cases (Negative Tests)
* System performs efficiently under expected load (Performance Tests)

The app’s modules include:  
Firebase Authentication, SurveyCTO Service, Parser, PDF Generator, FileManager/Storage, and End-to-End Workflow.

**6.2 Unit test cases**

**6.2.1 Firebase Authentication Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| UT-FB-001 | Login with valid credentials | 1. Open app 2. Enter valid email/password 3. Login | Login successful; session token returned |
| UT-FB-002 | Login with invalid credentials | 1. Enter invalid email/password 2. Login | Authentication fails; error displayed |
| UT-FB-003 | Login with empty fields | 1. Leave email/password empty 2. Login | Error prompting user to enter credentials |
| UT-FB-004 | Session persistence | 1. Login 2. Restart app | User remains logged in |
| UT-FB-005 | Fetch SurveyCTO credentials | 1. Login via Firebase 2. Fetch credentials | Correct SurveyCTO credentials retrieved securely |

**6.2.2 SurveyCTO Service Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| UT-SCTO-001 | Fetch forms list | 1. Call fetchForms() | Returns valid list of forms |
| UT-SCTO-002 | Fetch submissions | 1. Call fetchFormData(formId) | Returns correct submission data |
| UT-SCTO-003 | Missing credentials | 1. Call service without credentials | Error/prompt to provide credentials |
| UT-SCTO-004 | API/network failure | 1. Disconnect network 2. Call service | Proper error message; no crash |

**6.2.3 Parser Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| UT-PAR-001 | Parse valid JSON | 1. Provide valid JSON 2. Call parser | Correct Submission object created |
| UT-PAR-002 | Parse JSON with missing fields | 1. Provide JSON with missing keys 2. Call parser | Missing fields handled; defaults applied |

**6.2.4 PDF Generator Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| UT-PDF-001 | Generate PDF for single submission | 1. Pass one Submission object 2. Generate PDF | PDF created correctly |
| UT-PDF-002 | Generate PDF for multiple submissions | 1. Pass multiple Submission objects 2. Generate PDF | PDF contains all submissions properly |
| UT-PDF-003 | Template application | 1. Generate PDF 2. Check layout | PDF layout matches predefined template |

**6.2.5 File Manager / Storage Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| UT-FM-001 | Save PDF | 1. Call saveFile(pdfData) | PDF saved in local storage |
| UT-FM-002 | Share PDF | 1. Call shareFile(filePath) | PDF shared via platform |
| UT-FM-003 | Preview PDF | 1. Call previewFile(filePath) | PDF opens in preview window |

**6.3 Functional Test Cases**

These test whether the features work according to specifications.

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| FT-001 | Login via Firebase | 1. Open app 2. Login | Navigate to main dashboard |
| FT-002 | Fetch and display forms | 1. Login 2. Fetch forms | Forms displayed correctly |
| FT-003 | Fetch and display submissions | 1. Select form 2. Fetch data | Submissions displayed in correct format |
| FT-004 | Generate PDF | 1. Select submissions 2. Generate PDF | PDF generated with correct data |
| FT-005 | Save & share PDF | 1. Save PDF 2. Share PDF | PDF saved locally & shared successfully |

**6.4 Integration Test Cases**

These ensure modules work together seamlessly.

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| IT-001 | End-to-end workflow | 1. Login via Firebase 2. Fetch SurveyCTO credentials 3. Fetch data 4. Parse 5. Generate PDF 6. Save & Share | Workflow completes successfully; PDF correct |
| IT-002 | Handle expired SurveyCTO session | 1. Login via Firebase 2. Use expired credentials | App prompts to re-login or refresh credentials |

**6.5 Negative / Error Handling Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| NT-001 | Login with invalid credentials | 1. Enter wrong credentials 2. Login | Login fails; error message shown |
| NT-002 | Empty login fields | 1. Leave fields empty 2. Login | Error message shown |
| NT-003 | Missing SurveyCTO credentials | 1. Login via Firebase 2. Fetch data without credentials | Error displayed or prompt for credentials |
| NT-004 | Network failure during fetch | 1. Disconnect network 2. Fetch data | Proper error; app doesn’t crash |
| NT-005 | Generate PDF with invalid data | 1. Pass malformed Submission object 2. Generate PDF | PDF generation fails gracefully; error logged |

**6.6 Performance Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC ID** | **Description** | **Steps** | **Expected Result** |
| PT-001 | PDF generation with large dataset | 1. Fetch large number of submissions 2. Generate PDF | PDF generation completes within acceptable time |
| PT-002 | Multiple concurrent logins | 1. Login multiple users | All sessions handled correctly without crash |
| PT-003 | Data fetch under high load | 1. Simulate heavy API calls | App retrieves data correctly; no timeout |

6**.7 Notes**

1. **All tests were performed on the Flutter app**, with Firebase Authentication enabled and SurveyCTO credentials integration.
2. **Unit tests** focus on individual modules: FirebaseAuth, SurveyCTOService, Parser, PDFGenerator, FileManager.
3. **Functional tests** verify feature-level correctness.
4. **Integration tests** ensure smooth interaction between Firebase, SurveyCTO, and app modules.
5. **Negative tests** cover errors, invalid inputs, network failures, and session issues.
6. **Performance tests** ensure system handles large datasets and multiple operations efficiently.

**Chapter 7: Results & Discussion**

**7.1 Results**

* Successful integration with SurveyCTO API.
* Audit PDFs generated accurately from form data.
* Reports can be previewed, saved, and shared.

**7.2 Screenshots**

|  |  |  |
| --- | --- | --- |
| Logging in with test credentials | Filled forms and fetched data for hospital id and case id wise search | Filled forms and fetched data for hospital id and date wise search |
|  | | |
|  | | |
|  | | |
| Sample live audit pdf generation | | |

**7.3 Discussion**

The PM-JAY Live Audit App brings clear improvements over traditional methods. By **eliminating manual report writing**, it ensures that all audit findings are documented in a consistent, standardized PDF format. This reduces human error and improves the reliability of reports.

The system also **reduces audit processing time** by combining secure login, real-time data retrieval, and automated report generation into one workflow. This allows auditors to focus on analysis rather than repetitive tasks.

Moreover, the design is **scalable** and can be adapted for other healthcare audit systems or government programs that require structured data collection and reporting. With minor adjustments to templates and data mapping, the same framework can be reused in different contexts.

**Chapter 8: Conclusion & Future Work**

**8.1 Conclusion**

The PM-JAY Live Audit App successfully achieves its objectives of automating the live audit process. It ensures secure retrieval of SurveyCTO data, generates PDF reports, and provides easy sharing mechanisms. By leveraging Firebase Authentication, the app enhances user data security. In the future, Firebase Cloud Storage or Firestore could be used to store generated reports or audit metadata for centralized monitoring.

**8.2 Future Work**

* Offline mode for data storage and sync.
* Multi-language support for regional auditors.
* Integration with cloud storage (Google Drive/OneDrive).
* Admin dashboard for analytics.

**References**

[1] National Health Authority, Pradhan Mantri Jan Arogya Yojana (PM-JAY): Comprehensive Guidelines, Government of India, 2018. [Online]. Available: https://pmjay.gov.in

[2] A. Bhatnagar, A. Mishra, and S. Gupta, “Improving health service delivery through Ayushman Bharat,” Indian Journal of Public Health, vol. 64, no. 2, pp. 163–167, 2020, doi:10.4103/ijph.IJPH\_513\_19

[3] SurveyCTO, SurveyCTO Documentation, Dobility Inc., 2025. [Online]. Available: https://docs.surveycto.com

[4] C. Hartung, A. Lerer, Y. Anokwa, C. Tseng, W. Brunette, and G. Borriello, “Open Data Kit: Tools to Build Information Services for Developing Regions,” in Proc. 4th ACM/IEEE Int. Conf. Inf. Commun. Technol. Develop., 2010, doi:10.1145/2369220.2369236

[5] KoboToolbox, KoboToolbox User Guide, Harvard Humanitarian Initiative, 2025. [Online]. Available: https://support.kobotoolbox.org

[6] M. Tomlinson, M. J. Rotheram-Borus, L. Swartz, and A. C. Tsai, “Scaling Up mHealth: Where Is the Evidence?,” PLOS Medicine, vol. 10, no. 2, p. e1001382, 2013, doi:10.1371/journal.pmed.1001382

[7] C. Free, G. Phillips, L. Watson, L. Galli, L. Felix, P. Edwards, … A. Haines, “The Effectiveness of Mobile-Health Technology-Based Health Behaviour Change or Disease Management Interventions for Health Care Consumers: A Systematic Review,” PLOS Medicine, vol. 10, no. 1, p. e1001362, 2013, doi:10.1371/journal.pmed.1001362

[8] United Nations Department of Economic and Social Affairs, United Nations E-Government Survey 2022: The Future of Digital Government. [Online]. Available: https://publicadministration.un.org/egovkb

[9] H. Kaur and D. Singh, “E-Governance in Healthcare Sector: A Case Study of Ayushman Bharat,” Int. J. Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 7, pp. 2278–3075, 2019.

[10] Firebase, *Firebase Authentication Documentation*, Google LLC, 2025. [Online]. Available: <https://firebase.google.com/docs/auth>

[11] H. Almalki, A. S. Gray, and A. Sanchez, “Cloud Computing for Healthcare: Opportunities and Challenges,” *Future Internet*, vol. 13, no. 12, p. 319, 2021.