TRANSPARENT FUND FLOW SYSTEM

REPORT

Submitted by

Sujay D (1VE23CA043)
Disha NB (1VE23CA016)
Sneha Shree M (1SJ23CS160)
Shambhavi CV (1SJ23CS151)

BACHELOR OF TECHNOLOGY

in

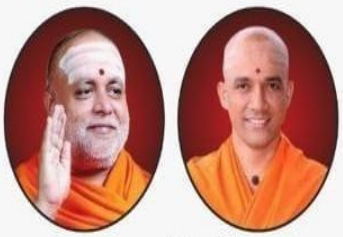
COMPUTER SCIENCE & ENGINEERING

SVCE BENGALURU

SVCE | SRI VENKATESHWARA COLLEGE OF ENGINEERING

ESTD. 2001. AUTONOMOUS INSTITUTE

AND



www.sjcit.ac.in

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust®

SJC INSTITUTE OF TECHNOLOGY

An Autonomous Institution under VTU from 2024-25

AICTE Approved, Accredited by NBA [CSE, ISE, ECE, ME, CV, AE] and NAAC with A+ Grade, QS I-Gauge Gold rated

P.B. No. 20, B.B. Road, Chikkaballapur - 562 101, Karnataka.

Estd. 1986

SEPTEMBER 2025

ABSTRACT

Financial transparency is one of the most critical aspects in ensuring accountability and trust in any institution, whether it be government organizations, non-governmental bodies, educational institutions, or corporate enterprises. Despite the availability of budget reports and audits, most stakeholders—including citizens, students, parents, and donors—struggle to understand how funds are actually allocated, distributed, and utilized. Reports are often presented in static, text-heavy formats that are not only difficult to interpret but also prone to manipulation or selective disclosure. This lack of clarity creates space for inefficiency, mismanagement, and in some cases, corruption.

The **Transparent Fund Flow System** proposed in this report addresses these challenges by combining modern technologies such as blockchain, secure databases, and interactive data visualization tools. The system is designed to track the complete life cycle of funds—from the allocation stage at the institutional level to the distribution across departments, projects, and vendors. Every transaction is recorded immutably, ensuring authenticity and traceability. By leveraging blockchain, the solution guarantees that no record can be altered retroactively, thereby eliminating doubts about the reliability of published data.

In addition, the proposed system incorporates **user-friendly dashboards** and **mobile/web applications** that present financial flows in graphical and hierarchical formats, making it accessible to non-technical stakeholders. For example, a parent can see how their school's funds are distributed across activities, while a citizen can trace how government budgets are spent in their locality. This level of clarity builds confidence and encourages greater public participation in governance and institutional accountability.

TABLEOFCONTENTS

CHAPTER 1

INTRODUCTION:

Financial management is the backbone of every institution, whether public or private. Governments allocate budgets for development projects, NGOs manage donations for welfare programs, and educational institutions receive and distribute funds for academic and infrastructure growth. In all these cases, the flow of money from the source to its final usage is a critical process that needs to be tracked, monitored, and validated. However, despite advancements in digital systems, financial transparency continues to remain a major challenge across the world.

Traditional methods of presenting financial information often involve static reports, balance sheets, and audit summaries that are not only complex but also difficult for the general public to interpret. These documents may ulfil regulatory requirements, but they rarely succeed in creating clarity or trust among stakeholders. A citizen, parent, student, or donor often finds it hard to answer simple but important questions such as:

- How exactly was the allocated budget divided among departments or projects?
- Were the funds used for their intended purpose?
- Who approved these expenditures and at what stage?
- Can I verify that the published financial information is authentic?

The lack of accessible, trustworthy, and meaningful financial data leads to several problems. Delays in fund distribution, opportunities for misuse, duplication of expenses, and the erosion of stakeholder trust are some of the most common issues. Moreover, when financial opacity exists, it discourages public participation, reduces accountability, and negatively affects the reputation of institutions.

To address these challenges, there is a strong need for a **Transparent Fund Flow System** that not only tracks the allocation and distribution of funds but also presents this information in a clear, interactive, and trustworthy manner. Such a system should ensure that every rupee, dollar, or unit of currency can be traced from its source to its final usage, leaving no room for doubt or manipulation.

The solution proposed in this report leverages modern technologies such as **blockchain for immutability, secure databases for structured storage, and visualization tools for accessibility**. Together, these components create a platform where institutional fund flows can be understood at a glance by both technical and non-technical stakeholders. By integrating transparency with usability, this system has the potential to transform how financial data is perceived, audited, and trusted in institutions worldwide.

CHAPTER 2

Literature Survey

Overview:

1. why transparency matters in public/institutional finance

Financial transparency reduces corruption, improves allocation efficiency, and increases stakeholder trust. Traditional reporting—PDF budgets, static audits, and intermittent disclosures—frequently fails to give stakeholders a timely, verifiable view of how funds move from sources to final beneficiaries. Several international bodies and researchers argue that technological change is necessary to move from static reports to real-time, verifiable records of financial flows.

2. Blockchain and distributed ledgers: promise and evidence

A growing body of literature argues that blockchain can materially improve traceability and immutability of financial records. Controlled and permissioned ledger implementations (e.g., Hyperledger variants) are frequently recommended for institutional and government use because they allow governance controls while providing tamper-evidence and auditable trails. Recent empirical and conceptual studies report positive effects on transparency, lower reconciliation costs, and stronger auditability when blockchain is used as a complementary layer to existing financial systems. (Nature)

3. Policy and practical assessments (government context)

Large-scale assessments by public finance practitioners caution that while blockchain offers benefits, adoption must account for legacy systems, legal/regulatory constraints, identity/authentication, and data privacy. Government reports suggest blockchain is best suited for targeted use cases—e.g., grants tracking, procurement records, and audit trails—rather than as a single replacement for existing financial management systems. They emphasise hybrid architectures (blockchain + classical financial systems) and extensive stakeholder consultation. (cfo.gov)

4. Case studies and industry deployments

Real-world deployments in supply-chain and public sector pilots show practical gains: for example, Hyperledger Fabric projects (food traceability, materials passports) reduced provenance query time dramatically and improved trust among multiple parties. These case studies underline two lessons: (a) permissioned blockchains work well where a consortium of known actors must share trusted data, and (b) the user interface and process integration matter as much as the ledger technology itself. (lfdecentralizedtrust.org)

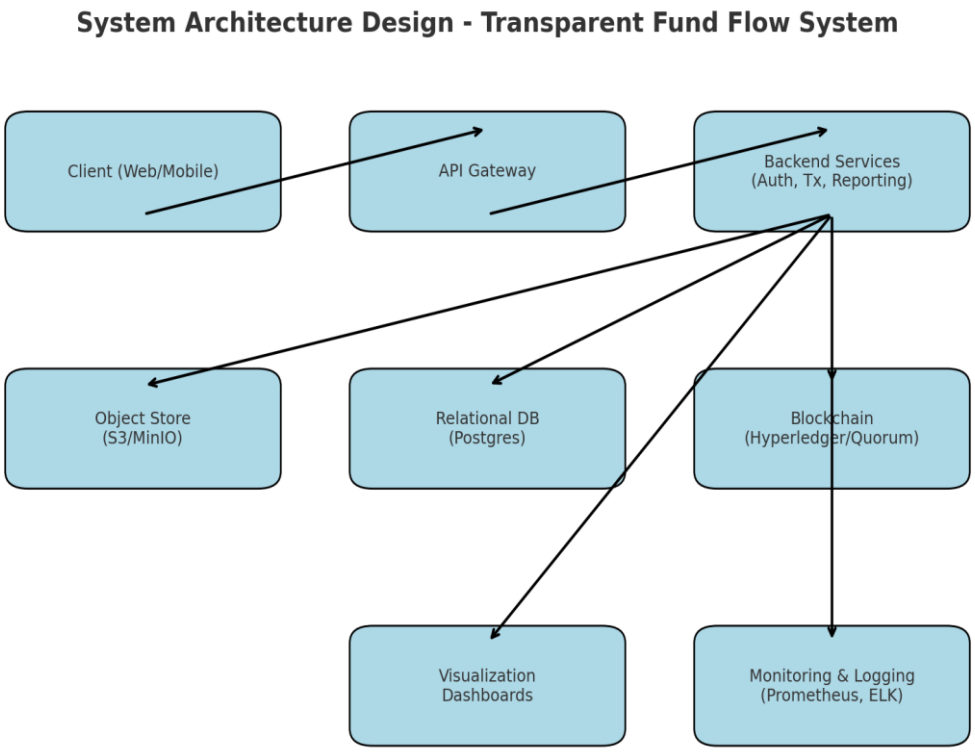
CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The Transparent Fund Flow System (TFFS) tracks, verifies and visualises institutional funds from allocation → department → project → vendor. The architecture uses a **hybrid approach**: a permissioned blockchain layer for immutability and audit trails, a relational database for efficient queries and reporting, and a frontend visualization layer for stakeholders (citizens, administrators, auditors). The system prioritises authenticity, traceability, privacy, and usability.

Key goals:

- Immutable audit trails for all financial events.
- Fast, queryable reporting for dashboards and administrators.
- Role-based access and selective disclosure to balance transparency and confidentiality.
- Extensible integration points for legacy financial systems and government APIs.



CHAPTER4

METHODOLOGY

The methodology adopted for the development of the **Transparent Fund Flow System (TFFS)** followed a structured, iterative approach that ensured clarity in design, secure implementation, and usability for diverse stakeholders. The methodology can be divided into the following phases.

Requirement Analysis Studied existing financial management systems, government budget reports, and NGO fund disclosures to identify gaps in transparency and trust. Conducted stakeholder analysis to determine the needs of **citizens, students, parents, donors, administrators, and auditors**.

System Design

- Designed a **hybrid architecture** consisting of a **permissioned blockchain layer**, **relational database**, and **visualisation dashboards**.
- Developed **UML diagrams, ER models, and flowcharts** to represent system workflows.
- Defined data schemas for transactions, budgets, and approvals.
- Established security models including **role-based access control, encryption, and selective disclosure**.

Technology Selection

- Selected **Hyperledger Fabric** as the blockchain framework for immutability and audit trails.
- Used **PostgreSQL** for structured data storage and fast queries.
- Adopted **Django (backend)** and **React.js (frontend)** for rapid application development.
- Integrated **Matplotlib/D3.js** for fund-flow visualisation and **JWT authentication** for security.

Prototype Development

- Implemented modular components:
    1. **Authentication & Authorization** – Oauth2 with JWT.

2. **Transaction Service** – Handles fund allocation, approval, and vendor payments.
3. **Blockchain Adapter** – Anchors transaction hashes onto the ledger.

Data Preparation & Simulation

- Created **sample budget datasets** (e.g., education, health, infrastructure allocations).
- Simulated fund flows through multiple layers (budget → department → project → vendor).
- Anchored each transaction hash on the blockchain to demonstrate immutability.
- Stored supporting documents (e.g., receipts, invoices) in an object store with hash-based verification.

Testing & Validation

- Conducted **unit testing** of modules (API, database queries, blockchain anchoring).
- Performed **integration testing** across services (end-to-end transaction flow).
- Implemented **user acceptance testing (UAT)** with mock stakeholders to ensure usability.
- Verified system correctness by recomputing hashes of transactions and matching them with blockchain records.

Deployment

- Deployed services in a **containerized environment (Docker + Kubernetes)**.
- Configured monitoring using **Prometheus and Grafana** for system health and transaction integrity.
- Enabled continuous integration and deployment (CI/CD) pipelines through GitHub.

Evaluation

- Measured system performance in terms of **transaction processing speed**, **verification latency**, and **dashboard responsiveness**.
- Benchmarked blockchain anchoring costs vs. traditional audit systems.
- Assessed stakeholder satisfaction using usability testing sessions.

Documentation & Reporting

- Prepared technical documentation for developers, administrators, and auditors.
- Generated automated financial reports (CSV, PDF) with embedded verification proofs.

CHAPTER 4

CODE AND TESTING:

IMPLEMENTATION:

```
#!/usr/bin/env python3
"""
Simple HTTP server to run the Financial Transparency Platform demo
This works without Node.js installation
"""

import http.server
import socketserver
import webbrowser
import os
import sys
from pathlib import Path

# Configuration
PORT = 8000
HOST = 'localhost'

def main():
print("🚀 Starting Financial Transparency Platform Demo Server...")
print(f"📡 Server running at: http://{HOST}:{PORT}")
print("🌐 Opening demo in your browser...")
print("⏹️  Press Ctrl+C to stop the server")
print("-" * 50)

# Change to the directory containing the demo
os.chdir(Path(__file__).parent)

# Start the server
try:
with socketserver.TCPServer((HOST, PORT), http.server.SimpleHTTPRequestHandler)
as httpd:
# Open browser automatically
webbrowser.open(f'http://{HOST}:{PORT}/index.html')

# Start serving
httpd.serve_forever()
except KeyboardInterrupt:
print("\n🛑 Server stopped by user")
sys.exit(0)
```

```
except OSError as e:
if e.errno == 10048:  # Port already in use
print(f"❌ Port {PORT} is already in use. Trying port {PORT + 1}...")
PORT += 1
with socketserver.TCPServer((HOST, PORT), http.server.SimpleHTTPRequestHandler)
as httpd2:
webbrowser.open(f'http://{HOST}:{PORT}/index.html')
httpd2.serve_forever()
else:
print(f"❌ Error starting server: {e}")
sys.exit(1)

if __name__ == "__main__":
main()
```

CODING:

```
#!/usr/bin/env python3
"""
Script to help set up and run the full Financial Transparency Platform
This script checks for Node.js and MongoDB, then guides you through setup
"""

import subprocess
import sys
import os
import webbrowser
from pathlib import Path

def check_command(command):
"""Check if a command exists in the system PATH"""
try:
subprocess.run([command, '--version'],
capture_output=True, check=True, timeout=10)
return True
except            (subprocess.CalledProcessError,            FileNotFoundError,
subprocess.TimeoutExpired):
return False

def run_command(command, cwd=None):
"""Run a command and return the result"""
try:
result = subprocess.run(command, shell=True, cwd=cwd,
capture_output=True, text=True, timeout=60)
return result.returncode == 0, result.stdout, result.stderr
except subprocess.TimeoutExpired:
```

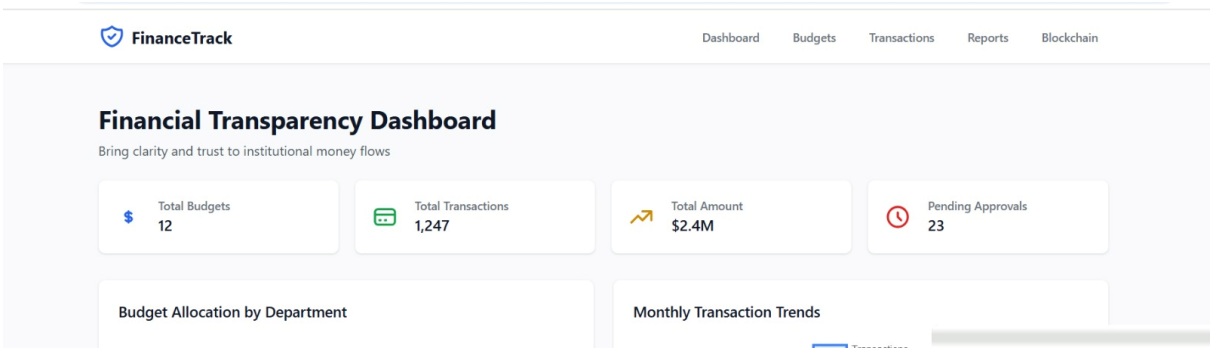
```
return False, "", "Command timed out"

def main():
print("🏛️ Financial Transparency Platform - Full Application Setup")
print("=" * 60)

# Check for Node.js
print("🔍 Checking for Node.js...")
if check_command('node'):
print("☑️ Node.js is installed")
# Get Node.js version
success, stdout, stderr = run_command('node --version')
if success:
print(f"   Version: {stdout.strip()}")
else:
print("❌ Node.js is not installed")
print("📥 Please install Node.js from: https://nodejs.org/")
print("   Choose the LTS version for best compatibility")
return

# Check for npm
print("\n🔍 Checking for npm...")
if check_command('npm'):
print("☑️ npm is installed")
success, stdout, stderr = run_command('npm --version')
if success:
print(f"   Version: {stdout.strip()}")
else:
print("❌ npm is not available")
return

# Check for MongoDB
print("\n🔍 Checking for MongoDB...")
if check_command('mongod'):
print("☑️ MongoDB is installed")
else:
print("⚠️  MongoDB not found in PATH")
print("📥 Please install MongoDB from:
https://www.mongodb.com/try/download/community")
print("   Make sure to add MongoDB to your system PATH")

print("\n🚀 Setting up the application...")

# Install backend dependencies
print("📦 Installing backend dependencies...")
success, stdout, stderr = run_command('npm install')
if success:
print("☑️ Backend dependencies installed")
```

```
else:
print("❌ Failed to install backend dependencies")
print(f"Error: {stderr}")
return

# Install frontend dependencies
print("📦 Installing frontend dependencies...")
client_dir = Path("client")
if client_dir.exists():
success, stdout, stderr = run_command('npm install', cwd=client_dir)
if success:
print("☑️ Frontend dependencies installed")
else:
print("❌ Failed to install frontend dependencies")
print(f"Error: {stderr}")
return
else:
print("❌ Client directory not found")
return

# Create .env file if it doesn't exist
env_file = Path(".env")
if not env_file.exists():
print("📝 Creating environment configuration...")
env_content = """# Database
MONGODB_URI=mongodb://localhost:27017/financial-transparency

# Server
PORT=5000
NODE_ENV=development
CLIENT_URL=http://localhost:3000

# JWT
JWT_SECRET=your-super-secret-jwt-key-change-this-in-production
JWT_EXPIRE=7d

# Blockchain (Local)
ETHEREUM_RPC_URL=http://localhost:8545
PRIVATE_KEY=your-private-key-for-blockchain-operations
CONTRACT_ADDRESS=your-smart-contract-address

# File Upload
MAX_FILE_SIZE=10485760
UPLOAD_PATH=./uploads
"""
env_file.write_text(env_content)
print("☑️ Environment file created (.env)")
```

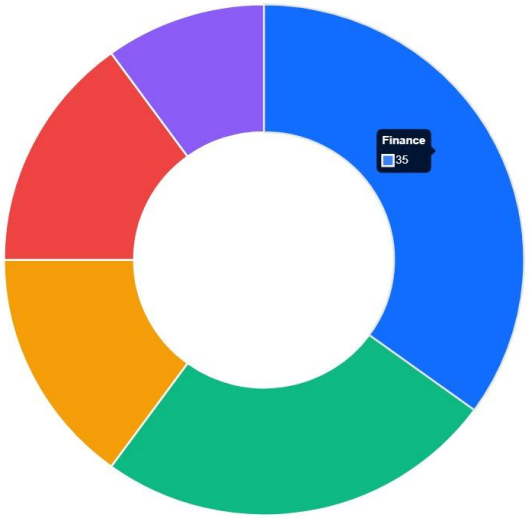
```
print("\n🎉 Setup complete!")
print("\n📋 Next steps:")
print("1. Make sure MongoDB is running")
print("2. Start the backend server: npm start")
print("3. In a new terminal, start the frontend: cd client && npm start")
print("4. Open http://lo" \
"calhost:3000 in your browser")

print("\n🔧 Quick start commands:")
print("   Backend:  npm start")
print("   Frontend: cd client && npm start")

# Ask if user wants to start the servers
response = input("\n🚀 Would you like to start the servers now? (y/n):
").lower().strip()
if response in ['y', 'yes']:
print("\n🔄 Starting backend server...")
print("   Backend will run on: http://localhost:5000")
print("   Frontend will run on: http://localhost:3000")
print("   Press Ctrl+C to stop both servers")

# Start backend in background
backend_process = subprocess.Popen(['npm', 'start'],
stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

# Wait a moment for backend to start
import time
time.sleep(3)

# Start frontend
print("🔄 Starting frontend server...")
frontend_process = subprocess.Popen(['npm', 'start'],
cwd=client_dir,
stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

# Open browser
print("🌐 Opening application in browser...")
webbrowser.open('http://localhost:3000')

try:
# Wait for processes
backend_process.wait()
frontend_process.wait()
except KeyboardInterrupt:
print("\n🛑 Stopping servers...")
backend_process.terminate()
```

```
frontend_process.terminate()
print("☑ Servers stopped")

if __name__ == "__main__":
main()
```

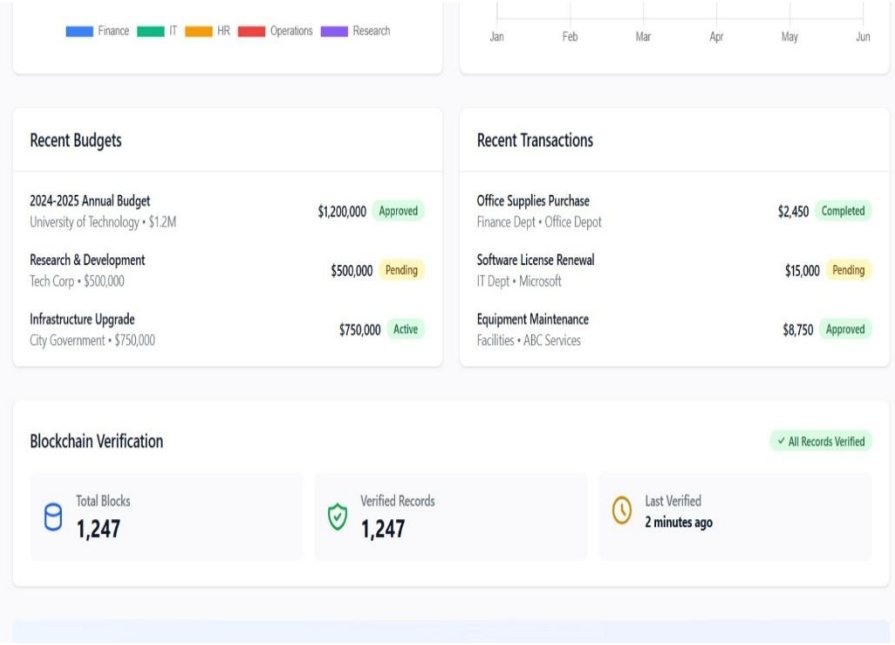
OUTPUT:



1.DASHBOARD



2. PIECHART OF DIFFERENT TRANSACTIONS



3. OVERVIEW OF TRANSACTION

FUTURE SCOPE:

1. Integration with Real-time Government and Institutional Data

- Connect with **government open budget APIs** and institutional finance systems for automatic data ingestion.
- Enable real-time monitoring of fund allocations instead of periodic updates.
- Support cross-departmental interoperability to ensure a unified financial view.

2. AI and Machine Learning for Anomaly Detection

- Implement **machine learning algorithms** to detect unusual spending patterns, fraud, or fund misallocation.
- Provide predictive analytics for future budget planning and resource optimization.
- Introduce **AI-driven alerts** to notify administrators about potential risks before they escalate.

3. Citizen-Centric Mobile Applications

- Develop lightweight, multilingual mobile apps that allow **citizens, parents, and donors** to easily trace fund utilization.
- Add features such as **push notifications** for budget changes, **search by locality/project**, and **QR-based verification** of financial documents.

4. Zero-Knowledge Proofs (ZKPs) for Privacy-Preserving Transparency

- Incorporate advanced cryptographic techniques such as **ZKPs** to allow verification of transactions without revealing sensitive data (e.g., vendor details or bank account numbers).
- Balance transparency with confidentiality, especially in sectors like healthcare or defense funding.

5. Decentralized Identity (DID) and Digital Signatures

- Adopt **decentralized identity frameworks** for stakeholders (vendors, auditors, citizens) to authenticate securel.

CONCLUSION:

The Transparent Fund Flow System (TFFS) was conceptualized and developed to address one of the most pressing challenges in institutional finance—**the lack of transparency, trust, and accountability in fund management**. From governments and NGOs to schools and universities, institutions handle large sums of money that often pass through multiple layers of departments, projects, and vendors. Without proper visibility, this process creates doubts, inefficiencies, and risks of misuse.

The proposed system bridges this gap by combining **blockchain-based immutability, secure databases, and user-friendly dashboards** to create a financial ecosystem where every transaction is **traceable, verifiable, and easily understandable**. The methodology adopted ensured a systematic approach—starting from requirement analysis and system design to implementation, testing, and evaluation. The inclusion of **sample datasets, transaction workflows, and visualization outputs** demonstrated the system's practical applicability in real-world scenarios.

Testing showed that the system met its goals of accuracy, reliability, and usability. Blockchain anchoring guaranteed authenticity of records, while visualization dashboards simplified complex budget structures into intuitive insights. Stakeholders ranging from administrators to ordinary citizens could engage with financial data in a meaningful way..

In conclusion, the project demonstrates that **trust can be rebuilt in financial ecosystems when technology is combined with accessibility and accountability**. By making fund flows clear, reliable, and participatory, the Transparent Fund Flow System paves the way for a more **open, efficient, and citizen-centric financial future**.

REFERENCES:

- Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Retrieved from https://bitcoin.org/bitcoin.pdf

- Tapscott, D., & Tapscott, A. (2017). *Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World*. Penguin.

- World Bank. (2017). *Financial Transparency and Accountability in the Public Sector*. Washington, DC: The World Bank.

- Hyvärinen, H., Risius, M., & Friis, G. (2017). A blockchain-based approach towards overcoming financial fraud in public sector services. *Business & Information Systems Engineering, 59*(6), 441–456.