ISSN: 2687-8097

Scientific Journal of Research and Reviews

DOI: 10.33552/SJRR.2026.04.000600

Iris Publishers

Research Article

Cloud Databased Contactless Mobile Payment System Framework: An Asymmetric Encryption Secured Financial Transactions Model

Hisham AbouGrad*, Andrey Shabarshov, Manasa Yegamati, Bilyaminu Auwal Romo and Mhd Saeed Sharif

Department of Computer Science and Digital Technologies, School of Architecture, Computing and Engineering, University of East London - UEL, London, United Kingdom

***Corresponding author:** Hisham AbouGrad, 4-6 University Way, Docklands, London, E16 2RD, UK.

Received Date: March 03, 2026

Published Date: March 16, 2026

Abstract

Contactless Mobile Payment Systems (MPSs) and financial digital platforms have become increasingly important to our societies due to the widespread use of mobile devices and smartphones. These MPSs and Financial technology (FinTech) platforms facilitate large-scale financial transactions and contribute to the decline of cash usage. Cloud database systems play a crucial role in enabling fast financial transactions over the internet using recognition technology. This research aims to deliver a secure framework for contactless monetary transactions by modelling an MPS based on QR code technology and a real-time database. The MPS software architecture includes a mobile application and a server-side application architecture to handle business logic. Users' data are digitally signed and stored using cloud computing technology. Agile methodology principles and the Scrum method are applied during system design, QRBee framework, and algorithms' development. The study experiments demonstrated that QR codes are suitable for monetary transactions and cryptographic methods, such as asymmetric cryptography algorithms and digital signature techniques, to ensure secure data exchange without requiring client-side internet connectivity in computer networks. The QRBee framework has been rigorously tested using an Android app with an API server and MongoDB real-time database. This shows an MPS software architecture and application toward a robust communication network technology model to ensure that people from anywhere can easily and securely process financial transactions.

Keywords: Cloud computing real-time database, Contactless mobile payment system, QR code technology, Agile software development, API network server, Asymmetric cryptography algorithms, FinTech Model

Introduction

Mobile Payment Systems (MPS) have become one of our essential daily life activities [1]. Cash and payment cards (e.g. credit cards, debit cards), as well as other payment methods (e.g. Gift cards) are still commonly in-use by many people [2, 3]. This paper addresses the growing need for secure contactless monetary transactions by proposing a system, which comprises a software system for Quick Response (QR) code generation and scanning and a server-side business management application [4, 5]. Also, the Agile methodology, Objectives and Key Results (OKRs) Framework, and the Scrum method are adopted for the research framework and software system development [6-9]. In addition, the .Net framework is used due to its cross-platform functionality, an integrated development environment, and the Xamarin library to develop an Android-based QRBee mobile app. This is supported by a quantitative approach using a questionnaire and data from related studies to gather the system requirements, design, and development.

The focus on QR code-based financial transactions revealed promising results from a public acceptance survey, as high confidence in the QRBee payment system has been observed [2]. The QRBee system is used in an Android-based mobile app and a three-tier architecture [10]. This showcases the suitability of QR codes for secure transactions. The research has established a secure platform for conducting contactless financial transactions to investigate the associated advantages, disadvantages, and areas for potential improvement [6-8]. A key distinguishing feature of the proposed system lies in its approach to financial transactions by utilising QR code recognition techniques and technology. This has been achieved by a comprehensive background review, which is conducted through exploring related topics from the current literature. The research methodology and methods focused on the design of the financial transactions system and protocol, followed by a review of its implementation. Results are detailed in the subsequent section by outlining the outcomes and the implemented algorithms.

The content of this paper provides a holistic on the research study, which is structured as follows: Section 2 presents the research context and literature review; Section 3 explains the research framework and applied methodology, methods, and techniques based on the literature to conduct the study and describe how the research has been implemented toward its findings; Section 4 presents the research findings including the application of QRBee framework and algorithms of the QRBee system; Section 5 comprehensively examines the research findings and explains the proposed system and further work, which includes responses to the research questions, outlines the study's limitations and validity, and concludes by discussing the potential implications; Section 6 ends the paper by drawing the research conclusions and areas for further work.

Background and State of the Art

In this section, the related work that has been published in the fields of contactless mobile payment systems, more specifically, their types will be reviewed. QR code structures and how they can be used in online transactions will be discussed. Also, a quick discussion about real-time databases will be presented. Finally, the combination of all these components will be outlined and explained to demonstrate how these can be used to process financial transactions.

Designing a Contactless Mobile Payment System

There are many mobile payment technologies that are currently in use to run our daily activities. The Ahmed et al. [2] study explored multiple proposed models of mobile payment systems, including their technologies, comparisons, payment methods, and different security mechanisms. This provided analysis of the encryption technologies, authentication methods, and firewalls, and also mentioned the following mobile payment technologies: Near Field Communication (NFC); GSM / SMS; Radio Frequency Identification (RFID); QR Code; Bluetooth; and Universal Second Factor (U2F).

Based on mobile payment technologies, different mobile payment systems have been developed. For example, the Yong, Rana and Shanmugam [3] study proposed a smart shopping system that enables customers to visit a shopping mall to scan the RFID of their purchase on the reader and watch its updates in real-time on their respective mobile devices. The hardware of the system contains Node32 Lite, which is a microcontroller, Mifare RFID Reader, Tag and Card. These will be the core components of the proposed system. The study used the following software development kits (SDKs): PHP programming language, PHPMyAdmin, as an administrative console, and Webhost for hosting their system. Also, the Nseir, Hirzallah and Aqel [4] study proposed QR codes and mobile phones to be used in a mobile payment system instead of credit cards and Personal Identification Number (PINs). QR codes have certain characteristics, such as 360-degree reading, resistance to distortion, data restoration, etc., which can be utilised in MPS [5].

Utilising Quick Response Code Structure

QR codes are a two-dimensional symbol that can store information just like their predecessors Barcodes. In recent years, QR codes have seen a spike in their popularity. The Soon's [5] study stated that a QR Code consists of the following 5 components:

- **Finder Pattern:** This is represented as three squares located in the three corners of the QR code. This pattern will allow the user to get the position, size, and angle of the QR code, which means that the QR code can be read from different angles.
- **Alignment Pattern:** This consists of one or more squares located at the same distance. The purpose of this pattern is to correct any distortions that QR code might have.
- **Timing Pattern:** This consists of repeating black and white cells arranged both vertically and horizontally. The timing pattern has the same purpose as the alignment pattern.
- **Quite Zone:** This is the zone that surrounds the QR code, and its purpose is to make QR code more recognisable.
- **Data Area:** The actual QR code consists of black and white cells. The data that is passed to the QR code is encoded into binary, which is then converted to black and white squares.

The Chang et al. [11] study focused on the motivation of customers to choose QR codes as a payment tool. The study applied the slightly modified model of Unified Theory of Acceptance and Use of Technology (UTAUT), where questionnaires were used for data collection, and more than 400 records were gathered, and also, Partial Least Squares Structural Equation Modelling (PLS-SEM) were used for data analysis. The study has found that potential customers would intend to use QR code in the following cases: First, if the customer perceived that using the QR code would improve transaction efficiency of other applications; Second, the money transaction cost would be reduced; Third, the customer can receive a discount by using QR code, and/or other people who are important to them are using QR code as well.

Ahmad et al. [12] introduced a payment system that uses QR code and visual cryptography to ensure maximum security. This system consists of two components, which are the mobile application and cloud server. The system's payment process is illustrated in Figure 1 and Figure 2.

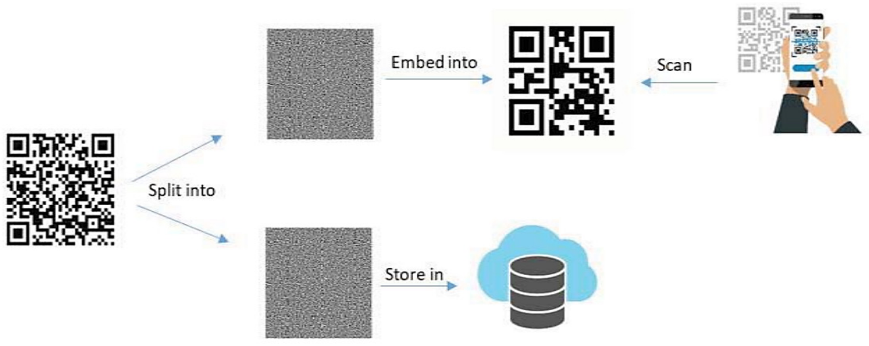


Figure 1: Process of Generating Shadows from a QR Code [12].

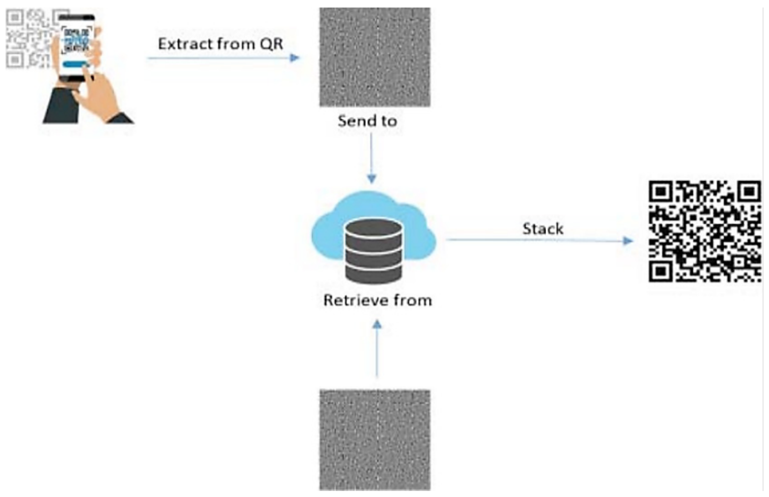


Figure 2: Process of Verification of QR Code [12].

The Visual Cryptography Scheme (VCS) algorithm has been used to conduct the implementation based on Ahmad et al. [12] study. The VCS algorithm generates two shadow parts to share, which are required to restore the original image. Figure 1 demonstrates the process of generating these two parts by sending a copy to the server and another copy to the user in order to scan and verify.

The QR codes are generated at the server's side for increasing security using image encryption and decryption to not allow any possibility of tampering at the client's side [12]. Also, the Sintyaningrum et al. [13] study demonstrated an application which can be used to enhance the security of certificates with digital signatures. This application can be used for extra security and a QR code for the authentication of the certificates. The study discusses in detail how their application works as well as presents some examples. Figure 2 shows the process of verifying the QR code upon scanning through two copies, which are a copy from the server and a second copy from the user who scanned the QR code.

Utilising Realtime Database

SQL database systems have been used for Online Transaction Processing (OLTP) since the emergence of the internet, but in modern database systems, NoSQL have gained more grip because NoSQL/unstructured databases provide better performance, especially in handling big data. In Ohyver et al. [14] study, NoSQL Firebase database engine compared to MySQL database, which results based on overall Create, Read, Update, and Delete (CRUD) operations that Firebase Realtime Database shows better performance relatively than MySQL Database. Conversely, the Kaur and Sachdeva [15] study introduces the NewSQL term as follows: "NewSQL is a class of modern Relational Database Management Systems (RDBMS) that provide the same scalable performance of NoSQL systems for Online Transaction Processing (OLTP) read-write workloads while still maintaining the ACID guarantees of a traditional database system". The work compares the performance of VoltDB, MemSQL, CockroachDB, and NuoDB. Figure 3 shows the comparison of NewSQL performance results.

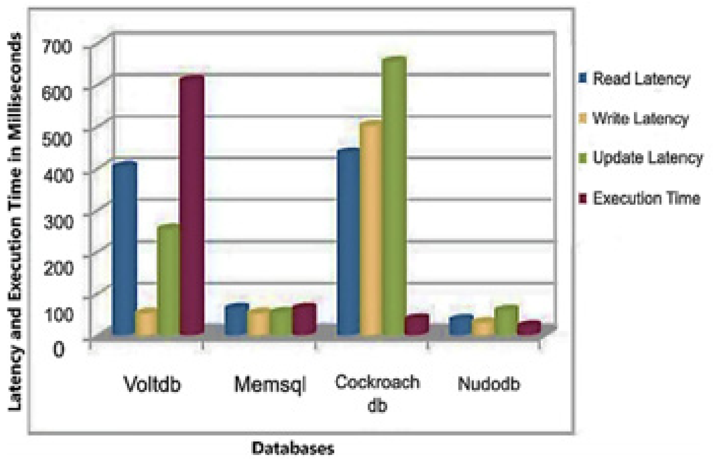


Figure 3: NewSQL Performance Comparison [15].

Further, traditional relational databases, which use Structured Query Language (SQL), are also developing [16]. For example, Oracle introduced Database In-Memory (DBIM) to describe and analyse the combination of DBIM and Active Data Guard (ADG) performance for a traditional Oracle database. The ADG provides nearly real-time primary-to-standby database replication to allow the offload of analytical tasks to the standby database instead of running them on a primary database, while DBIM functionality is extended to a standby database. Accordingly, developers can utilise the best of both frameworks by isolating their read-write and read-only workloads on the primary and standby databases to perform better and achieve faster analytics [16]. Indeed, DBIM-on-ADG structure allows making queries executed on the standby database to gain the benefits of DBIM for improving the response time of certain queries by orders of magnitude.

Developing a Contactless Mobile Payment System

The development of a robust and secure financial transaction processing system requires a mobile application (i.e., Mobile Payment System), which generates, scans, and reads QR codes. This uses cryptographic algorithms to formulate QR codes for protecting financial data. The mobile application processes financial transactions through a server where the MPS real-time database is allocated to perform all the required checks. Also, the MPS real-time database can store financial data and information about financial transactions. Overall, mobile applications development methods, techniques, and Software Development Kits (SDKs) have been used to clarify and make responses to the following: What exact data formats to use? What data streams should be involved? How will financial data be protected from accidental or fraudulent disclosure? What exactly should be stored in the database, and how to deal with unfinished transactions? Thus, software development methods, such as Agile Software Development (ASD) approach and Objectives and Key Results – OKRs framework, and SDKs (i.e., Microsoft Visual Studio, C# for the .Net, Xamarin framework, and MongoDB) have been used to create the process to develop the MPS software application and its real-time database [6-8]. These are discussed in the following sections to provide further details on the software application development and practice to formulate contactless MPS using QR codes.

Methodology and Methods

The development of the proposed contactless mobile payment system using QR code structure and real-time database integration was carried out following the principles of the Agile methodology, with a specific focus on the Scrum framework [10, 17]. Agile was selected as the guiding approach due to its iterative and incremental nature, which enables flexibility in addressing the continuously evolving requirements, continuous feedback, and collaborative development throughout the lifecycle of the project. The Scrum framework facilitated efficient sprint planning, daily stand-ups, sprint reviews, and retrospectives by ensuring the development process remained adaptive and responsive to challenges encountered during implementation. The MPS framework was implemented using the C# programming language within the .NET framework, as this environment provides cross-platform functionality, robust performance, and an extensive standard library that simplifies software development. The Visual Studio 2022 Community Edition served as the integrated development environment (IDE), which offered advanced debugging tools and seamless integration with version control systems to support efficient coding and testing. The .NET 6.0 framework and C# 10 were deliberately chosen to ensure compatibility with the latest technological advancements, while also utilising improved runtime performance and modern programming constructs.

Since the project was developed from the ground up, without any pre-existing legacy requirements, the latest available versions of tools and frameworks were utilised. This approach not only ensured the adoption of up-to-date development practices but also minimised potential compatibility issues in the long term. The incremental nature of the Agile-Scrum methodology allowed for the continuous refinement of the mobile payment system, particularly in implementing secure QR code generation, real-time transaction validation, and robust communication with the backend database.

The research study considered several software development methods prior to starting the actual system design and implementation. This includes studying their advantages and disadvantages (pros & cons) to find the most appropriate software design approaches and development methodology. Indeed, many requirements have been identified during the initial stage of the study, which include the following key requirements:

- Capability to recognise, scan and read the information embedded in the QR codes;
- Ability to make financial transactions between two client's mobile devices;
- Ensure that the designed model process tested on actual mobile devices in such different real-world scenarios (study cases).

Further, the research questions in which utilised to drive our study and the approaches and methods used are explained in Section 3.1, and in Section 3.2 the study direction has been described, including the search process, selection steps, and data sampling. In Section 3.3, the research design has been explained with details on the study methodology, methods and techniques toward making the research framework. Finally, Section 3.4 explains the software architecture and how the QRBee system utilises a cloud-based real-time database to ensure efficiency and adaptability for contactless monetary transactions.

Research questions

The main goal of this study is to research the feasibility and possibility of utilisation of QR code-based segments within secure and safe communication networks that are used for processing sensitive financial data [2-4]. This raises three Research Questions (RQs), which the study focuses on to provide the results and research findings, and these research questions are:

RQ1. What software development methodology and application lifecycle should be applied? During the initial phase of the study, a wide variety of software development methodologies were considered for implementation in this study. After careful investigations using related studies from the literature review, two main approaches have been considered, which are the Waterfall and ASD methodologies [6-8]. The aim is to select the most flexible software development approach in order to conduct the system design and implementation.

RQ2. Can QR Code be utilised as a transport tool for financial data? Usually, QR code do not provide fully secure communications. Simply because the QR code is visible to the naked eye. Similarly, many studies have shown that QR code is already commonly used by MPS [5, 15]. Hence, research should be conducted to define where in the communications chain such a not fully secure segment can be used (RQ2.1), and how QR code can be used to provide fully secure transport for financial data (RQ2.2). Also, the study needs to define the appropriate dataset to conduct the implementation of the transaction protocol, which specifies the stages for performing the payment. Thus, the study seeks to find how the dataset should be defined for each stage (RQ2.3).

RQ3. What are the key techniques to ensure the integrity of secure financial transactions? Certainly, one of the key objectives of this research study involves financial data, which must be measured appropriately to ensure that the financial transaction is genuine. Usually, data required for performing financial transactions, and therefore, a set of techniques has been applied to ensure the integrity and safety. Thus, the study looks at how to ensure the genuineness of financial transactions (RQ3.1). Also, it is important to protect data by following the appropriate measures in all stages of transaction to prevent tampering with the financial data and verify the origin of such data, so responses can be created accordingly. Thus, the study seeks to find how to recognise the appropriate measures to prevent access to genuine financial data to protect financial transactions (RQ3.2).

Study Direction

The focus of this paper is to deliver a FinTech model that can support monetary transactions supported by QR codes. As a result, systematic research was conducted as illustrated in Section 2. Also, related work is found mainly in three scholarly resources, which are IEEE Xplore database, Google Scholar, and Elsevier. This is achieved with the use of the University of East London library resources to find the related research materials.

Search and Selection Process

The primary search was conducted during 2022 using the above learning resources. A set of keywords were used during the search, which are: Mobile Payment Systems; QR Code; Realtime Database System; NoSQL Database; Online Transaction Processing; and Financial Technology. Only papers written since 2018 were considered relevant, and this filter was applied during the search. The combined results in the IEEE database returned around 13,967 results, of which the first 200 were reviewed. The Google Scholar results returned more than 200,000 papers (i.e. 262,900), of which the first 200 were also reviewed. In total, around 150 related research papers were gathered in an initial search.

Study Selection Steps

Selection steps were applied for the selected papers from the initial research and the above selection process. Firstly, the relevance of each identified study was first checked based on the title and the abstract, and any duplicates were removed. The remaining papers were read in their entirety. During the selection, the following criteria were applied to each paper: it must be written in English; it must be related to the selected methodology and the research framework; and it must be related to the selected topics or keywords. Lastly, almost 30 related works were used, along with the references from these studies, which also helped to further indepth our knowledge. Of course, these studies and resources are used in this paper's referencing.

Data Collection and Selected Studies

Research quantitative approach using close questions type of questionnaire (survey) chosen to collect data for the software model requirements [2]. The questionnaire aims to find the boundaries of comfort usage of mobile payment applications based on QR code and the advantages and disadvantages of QR code-based apps compared to other payment methods. The collected data were stored in a shared drive and processed. The survey was created using Microsoft Forms and shared in a Microsoft Teams group chat with students at the university. The survey consists of 12 questions represented as either Likert charts or single-choice questions, and the participant can only choose one of these answers. The research datasets are accessible through the Digital Commons Data (Mendeley Data) repository [18]. This is explained in section 4, and in section 5, the study findings and contributions have been illustrated.

Study design

Software development methodology is usually applied to ensure that a set of principles is followed as guidance during the study or software project to design the required models and system strategy diagrams [6-8]. According to Alliance Software [19], Software Development Methodology runs as a framework, process, or series of processes, which can be applied for software development purposes through different phases with steps to launch a software product. Also, the objectives and key results approach, also known as the OKRs framework, is implemented to provide regular checks on priorities, focus on achievements, and allows us to collaborate, communicate and together align tasks [7]. According to Doerr and Page [20], OKRs is a simple goal setting framework using Why? as a concept. OKRs are about Yes, no, and being Simple, which supports focusing on execution and what really matters to measure performance, as shown in Figure 4. Many organisations, such as Google, Amazon, Spotify, Twitter, and Netflix, use OKRs as an instrument to set and align their goals and define their desired results.

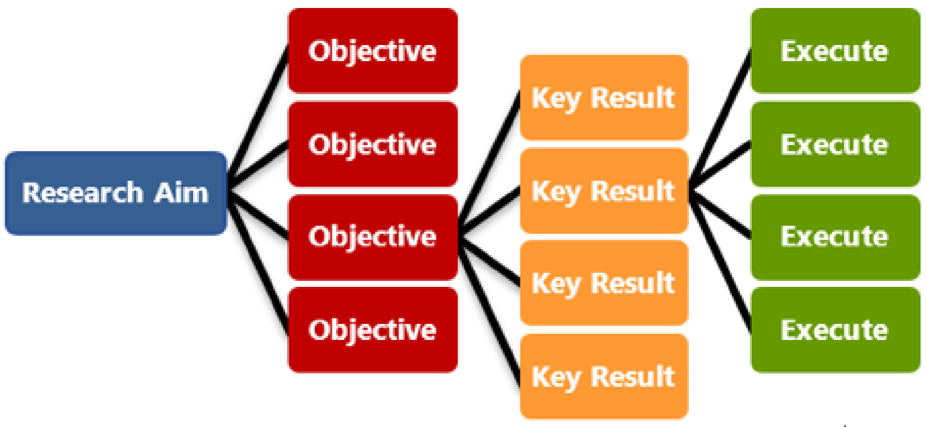


Figure 4: The Objectives and Key Results – OKRs Framework [20].

Further, the study ideas are executed using Agile software development. ASD presents a collection of several software development approaches based on similar ideas [8, 17]. These ASD approaches can organise the software development to embrace requirements, tackle software development changes, and prioritise tasks. Also, ASD focuses on agility and flexibility to formulate software products. Indeed, Agile is a set of principles based on the so-called "Agile Manifesto", which is an ideology to make a system of ideas including: Individuals and interactions over processes and tools; Working software over comprehensive documentation; Customer collaboration over contract negotiation; and responding to change over following a strategy and plan [21].

Practically, software developers and system architectures have multiple Agile methods, which are used to support and complement the Agile philosophy. Agile methods, such as Test-Driven Development (TDD), Feature Driven Development (FDD), Extreme Programming (XP), Scrum, Dynamic System Development Model (DSDM), and Crystal, etc., in general have their advantages and disadvantages, life cycles, and roles [22]. Indeed, there are several methodologies built using principles of Agile in mind, and one of them is formally known as Professional Scrum Framework (PSF) and generally as Scrum, see Figure 5 for details of the PSF process [21-23]. According to Scrum.org [9], Scrum has been described "as a way to get work done as a team in small pieces at a time, with continuous experimentation and feedback loops along the way to learn and improve as you go".

Further, Scrum (PSF) is applied as an iterative framework consisting of short iterations called "Sprints" [9, 21]. At the end of each sprint, the team delivers their work and measurable results. This helped the team to have constant collaboration with stakeholders and continuous improvement at each stage, which is vital to the project team members and stakeholders [17, 23]. Once each sprint begins, the team cycle through a process of planning, execution, and evaluation.

Software Architecture

The Financial Transactions Model (FTM) has been designed based on three layers, which are the mobile application, Application Programming Interface (API) server, and the database, see Figure 6 [24]. The mobile application layer is used to make transactions for both the client (payer) and the merchant (trader) actors [25]. The API server layer is used to manage the application functionality and make the required database queries using NoSQL database engine, and the NoSQL database layer stores transactions and users' data [26-28].

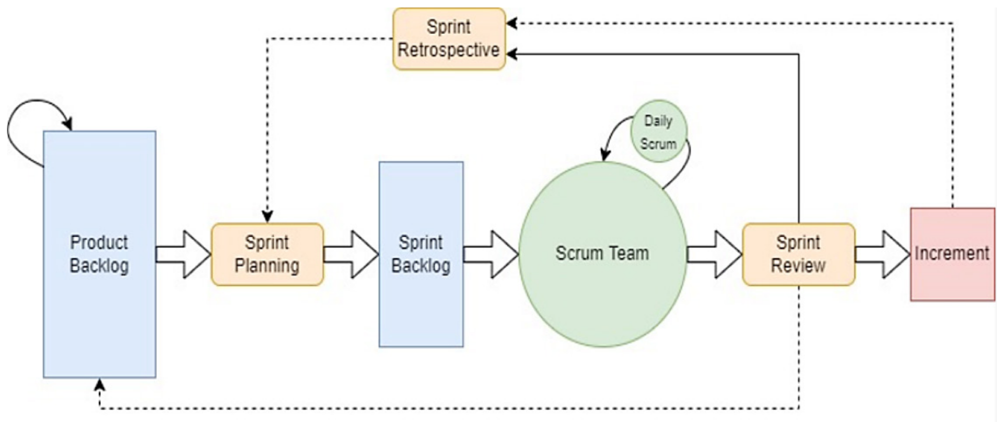


Figure 5: Professional Scrum Framework [9,21].

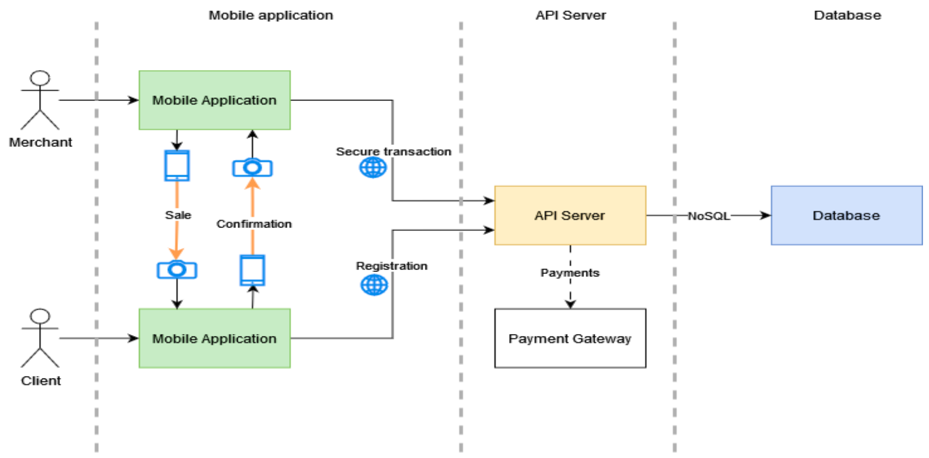


Figure 6: Financial Transactions Model Layers and Data Flow Diagram.

Both Client and Merchant interact with the mobile application model that functions using their role (Figure 6). Hence, the Client can register by accessing the API server, and then the transaction can be confirmed without internet access to the API server. While Merchant can register, initiate a transaction, receive a confirmation from the Client, communicate with the API server, and receive approval of the transaction. The mobile application model generates and scans QR codes to perform secure communication through a server using Microsoft Xamarin and C# for the .Net and the API server handles the connection logic [10]. NoSQL MongoDB database engine is used for the connection logic due to its superior performance, resilience, and comprehensive reporting capabilities [26]. External libraries for the mobile Operating Systems (OSs) have been contained in an object called QRBee to build the FTM dependency diagram (Figure 7). These libraries make the payment system have cross-platform capabilities. The FTM consists of five modules, which are:

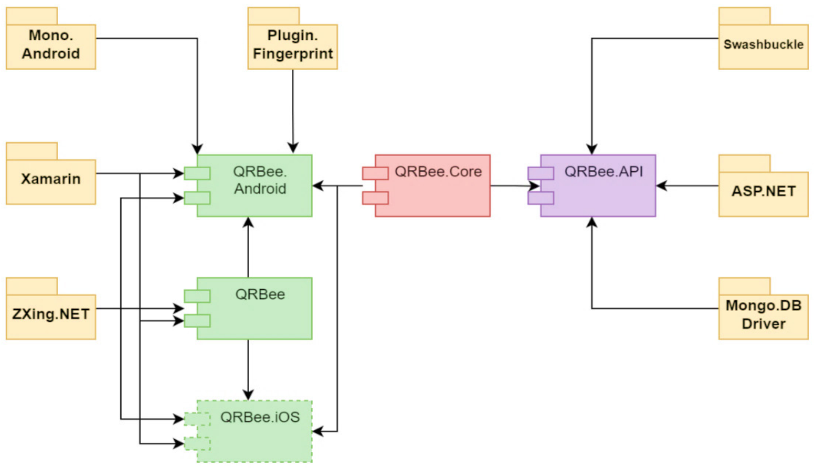


Figure 7: Financial Transactions Model Dependency Diagram.

- the QRBee.API for the API server that contains the connection functionalities;
- the QRBee.Core for the common functionalities;
- the QRBee.Android for Android OS and User Interface (UI) related functionalities;
- the QRBee for the common functionality for all mobile platforms;
- the QRBee.iOS for Apple iOS and UI related functionalities.

There are two main Client/Server communications implemented, see Figures 8 and 9. The first is the Client registration process to use the application, see Figure 8. The registration process is done with the use of private and public key encryption and self-signed certificates [29, 30]. For security reasons, private key generation with the use of a cryptographically strong random number generator must be performed at the point of ownership. Also, user (Client/Merchant) keys should be generated on the app, while server keys must be either pre-generated or generated on the API server-side [31]. The API server acts as a Central Authority (CA), which issues self-signed digital certificates to users. The user certificate is used during the registration validity procedure, which only accepts certificates issued by the API server and assigns them to the users' private key. Accordingly, if the user already exists, the certificate will be updated on the app; otherwise, the user's self-signed certificate will be stored in the real-time database (MongoDB).

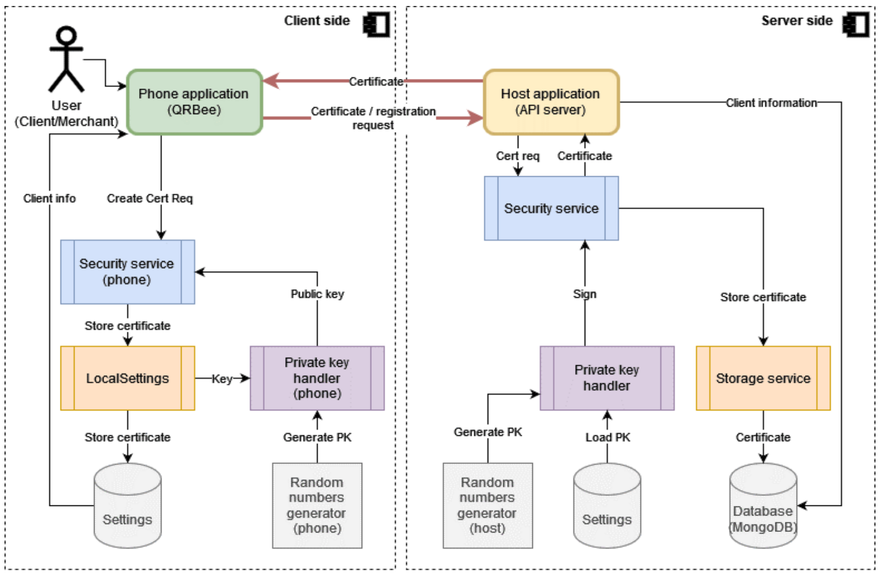


Figure 8: Financial Transactions Model Client Registration Graph.

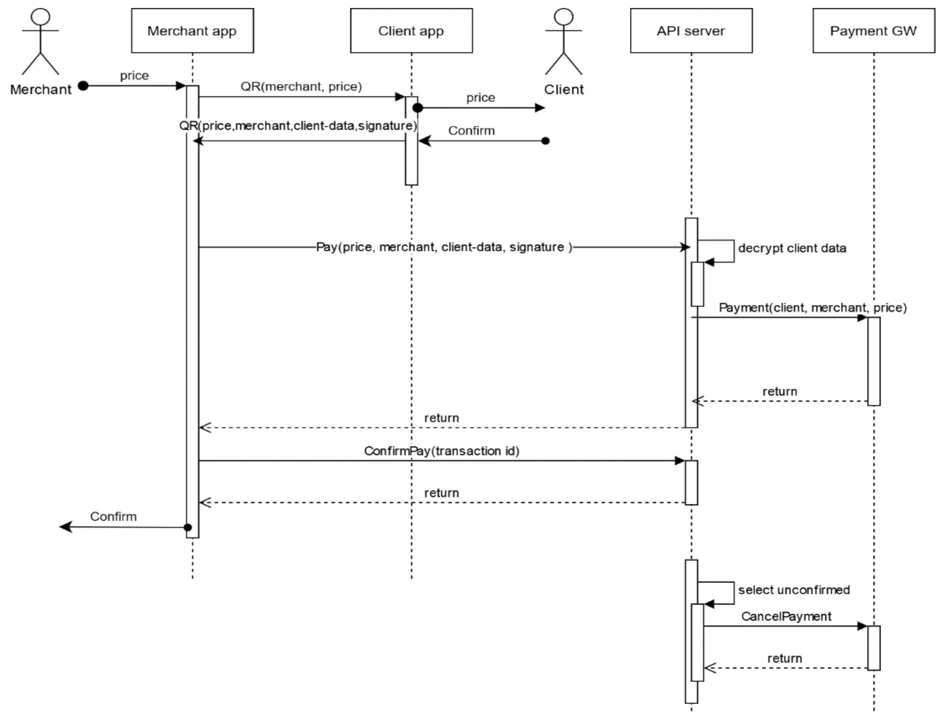


Figure 9: Financial Transactions Model Payment Sequence.

Table 1: Financial Transactions Model Sequence Steps.

№	Step	Description
1	Merchant generates QR code through mobile device	The user acting as a Merchant generates a QR code on his phone with the amount he/she wants the client to pay. The generated QR code consists of Merchant and transaction IDs, Merchants' Name, amount to be paid (money), the Coordinated Universal Time (UTC) Timestamp and Merchants' digital signature for the above.
2	Client scans QR code	The client uses the phones' camera to scan Merchants' QR code.
3	Client either accepts or denies the offer	If the client denies the offer, the transaction process will be ended, and if accepts, a new QR code is generated containing Merchants' data from step 1 with encrypted Client's card data, ID, timestamp and digital signature.
4	Merchant scans Client's QR code	The Merchant scans the Clients' QR code using Mobile camera.
5	Sending Merchant's data to the API server	The payment request only contains Clients' response data, which is sent to the API server for validation.
6	API server verifies the Merchant request	The verification is performed through a series of checks to validate the transaction data including digital signatures and Client cards' data in the system. Also, decrypts the Client's card data block.
7	Validated data sent to a payment gateway	A preliminary transaction is saved to the real-time database (MongoDB). The Client cards' data sent to the payment gateway, which rechecks data again and either accepts or rejects the transaction. The result gets converted into a transaction response and sent back to the Merchant
8	Merchant confirms the receival	The confirmation process happens in the background and it is intended to prevent the situation where the transaction gets accepted, but the API server fails to send the response back to the Merchant's mobile device.

The second Client/Server communication is the payment transaction process (Figure 9). The process uses a Two-phase commits technique [32]. This process consists of 8 steps, which are described in Table 1.

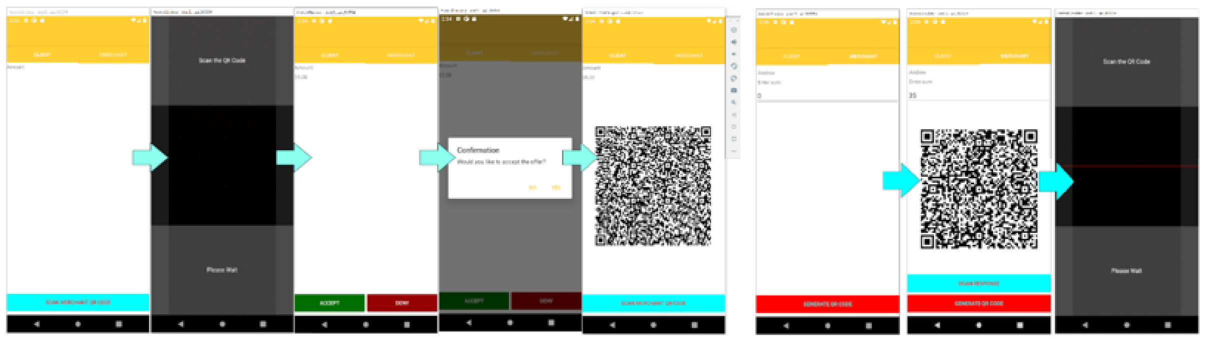
The Merchant is unable to access Clients' card information at point in the transaction process. The Client's card data block is fully encrypted, so that neither card number nor Card Verification Value (CVV code) are visible to other users. The encrypted data can only be decrypted by the API server. In parallel with the payment processing, a transaction cancellation loop runs in the background. The purpose of such loop is to automatically cancel unconfirmed transactions by Merchant's in a specified timescale, and the completed transactions are saved in the database.

Discussions and Findings

The study results are described in this section. The study seeks to assess the feasibility of developing a monetary transaction system that utilises QR codes. Section 3.2 explains a brief survey conducted to evaluate public acceptance of the proposed concept. Also, discussions about the proposed system's benefits to people are developed. This section looks closer at the user interface of the proposed system, its usability and User Experience (UX), and also provide an overview of the experiments on the system and their results. Finally, this section concludes by answering the research questions in section 3.1.

The purpose of QRBee is to provide small businesses and individuals with a system that secures contactless payments. To confirm this, a brief survey was conducted to find the boundaries of comfort usage of mobile payment based on QR code and the advantages and disadvantages of QR code compared to other payment methods. The results of the survey and responses from people during the project showcase event at the University of East London confirmed that 90% of participants would trust such a new payment system, as it provides a new, secure way of payment. Another study [33], also confirmed that QR code mobile payment (m-payment) experience have optimistic perspective and improved technology-based financial services.

The QRBee framework is implemented on an Android mobile app for experiments, and the main use interface consists of two tabs, Client and Merchant (Figure 10). Each tab plays its own role and corresponds to the transaction based on the FTM sequential steps. Figure 10.a shows the Client tab functionalities, and Figure 10.b shows the Merchants tab functionalities. In practice, the mobile phone camera should be enabled/allowed for the QRBee model to proceed, and then, register into the QRBee framework by clicking the "Register" button.



(a) Client tab functionalities (b) Merchant tab functionalities

Figure 10: QRBee framework mobile application functionalities.

A financial transaction model was implemented and tested (Table 1). This FTM has sequential steps based on algorithms, which use QR codes to store encrypted data for a financial transaction, and the FTM steps were discussed in section 3.4. The FTM has three algorithms to implement the model's sequential steps to process financial transactions [34]. Algorithm 1 describes the Merchants' steps 1, 4, 5, and 8 of the FTM sequence process (Table 1), and also, the Merchants' registration process and digital certificate assignment, if there is none in the QRBee framework.

Algorithm 1: Merchant

```
Input: price
Output: confirmation
1.  TryLoad MerchantCertificate, ServerCertificate
2.  if MerchantCertificate is None:
3.      show Merchant registration dialog
4.      MerchantKey: = new (public key, private key)
5.      MerchantCertificateRequest: = new CertificateRequest (MerchantKey.public, MerchantId)
6.      ServerCertificate, MerchantCertificate: = send MerchantCertificateRequest
7.      store MerchantCertificate, MerchantKey.private
8.  end if
9.  MerchantTransaction: = Sign (Make (Price), MerchantKey.private)
10. show QR Code (MerchantTransaction)
11. ClientTransaction: = read QR Code
12. ServerReply: = send ClientTransaction to API server
13. Check signature (ServerReply)
14. if signature is invalid
15.     stop
16. end if
17. MerchantConfirmation: = Sign (Make(transactionId), MerchantKey.private)
18. send MerchantConfirmation to API server
19. show confirmation
```

Algorithm 2 describes the Clients' steps 2 and 3 of the FTM sequence process (Table 1) and looks at the Clients' registration process and digital certificate assignment, if there is none in the QRBee framework. When the Client scans the Merchants' QR code, the FTM checks Merchants' signature and replies to the offer, and if either the Merchants' signature is incorrect or the reply is negative, algorithm 2 terminates the financial transaction.

Algorithm 2: Client

Input: QR Code (merchant transaction)

Output: QR Code (client transaction)

```
1.  TryLoad ClientCertificate, ClientCardData, ServerCertificate
2.  if ClientCertificate is None:
3.   ClientCardData: = show client registration dialog
4.   ClientKey: = new (public key, private key)
5.   ClientCertificateRequest: = new CertificateRequest (ClientKey.public, ClientId)
6.   ServerCertificate, ClientCertificate: = send ClientCertificateRequest to API server
7.   store ClientCertificate, ClientKey.private
8.  end if
9.  MerchantTransaction: = read QR code (merchant)
10. Check signature (MerchantTransaction)
11. if signature is invalid
12.  stop
13. end if
14. reply: = show price to client
15. if reply is OK:
16.  EncryptedCardData: = Encrypt (ClientCardData, ServerCertificate)
17.  ClientTransaction: = Sign (Make (MerchantTransaction, EncryptedCardData), ClientKey.private)
18.  return QR code (ClientTransaction)
19. else
20.  stop
21. end if
```

Algorithm 3 represents the API servers' steps 6 and 7 of the FTM sequence process. In this algorithm, the API server checks the Client's transaction data, including Merchants' and Clients' digital signatures, and if the transaction is appropriate, the data will be sent to the payment gateway, which responds back to the Merchant to confirm the transaction passes the check, so the transaction can be saved to the database with the "confirmed" status.

Algorithm 3: API server

```
Input: ClientTransaction
Output: ServerReply
1.  Check client signature (ClientTransaction)
2.  if signature is invalid
3.      return ServerReply (error)
4.  end if
5.  Check merchant signature (ClientTransaction.MerchantTransaction)
6.  if signature is invalid
7.      return ServerReply (error)
8.  end if
9.  ServerTransaction: = Make (ClientTransaction, Status: pending, timestamp)
10. store ServerTransaction
11. reply: = send ServerTransaction to Payment Gateway
12. if reply is OK
13.     ServerTransaction.Status: = unconfirmed
14.     store ServerTransaction
15. else
16.     ServerTransaction.Status: = declined
17.     store ServerTransaction
18.     return ServerReply (declined)
19. end if
20. ServerTransaction: = Sign (ServerTransaction)
21. confirmation := send ServerTransaction to merchant
22. if confirmation is OK
23.     ServerTransaction.Status := confirmed
24.     store ServerTransaction
25. end if
26. foreach ServerTransaction where Status is unconfirmed
27.     if ServerTransaction.timestamp < now - ConfirmationTTL
28.     ServerTransaction.Status := timeout
29.     store ServerTransaction
30.     send ServerTransaction to Payment Gateway
31.     end if
32. end foreach
```

Further, the Client needs an internet connection to register in the QRBee system during the registration process. The implemented financial transactions algorithm permits only the Merchant to have a constant internet connection, as the Merchant can only interact with the API server during realtime connectivity. The QRBee system uses an industrial standards three-tier architecture, which isolates the database from business logic, and business logic from the UI. The payment transaction was successfully implemented using QR codes on the Merchant/Client side and internet access on the Merchant/API Server side (RQ1). Experiments were conducted using an Android emulator and realtime mobile devices. The experiments revealed that Android emulator camera support was not ideal, as it has a very slow frame rate and resolution, which affects QR code recognition, especially if the QR code contains a high amount of data. The distance to the emulator camera also affected the recognition rate. As shown in Section 3.4, QR codes used within the application contain relevant data where the digital signature is the main size contributor. This means that QR codes are well suited to be a medium of monetary transactions (RQ2).

Furthermore, the proposed protocol provides a cryptographic method of data exchange (RQ3). This includes asymmetric cryptography, digital signatures and proof of key ownership, which provides a private certificate exchange scheme, see Section 3.4. At any point of data exchange, a digital signature is calculated based on the previous stage data to confirm the financial transaction. This makes data tampering impossible. Indeed, the only source of trust in the QRBee system is the API server certificate, which is distributed along with the mobile application. This is protected by Android or iOS developer signature and a corresponding digital certificate.

Overall, the QRBee system and its mobile application were developed using industry-standard practice, which utilised a three-tier architecture through Android OS. The protocol used by the QRBee system provides a robust and secure way of performing monetary transactions where its security relies on cryptographical transformations and digital signatures. QR codes were used for transaction initiation without an internet connection required on the Client side during payments, which makes financial transactions symbol through less requirements.

Experimental Analysis

This section discusses the research findings and provides explanations about the proposed system and makes recommendations for further development. This includes answering the research questions and providing the limitations and validity of this study. Finally, the implications for potential research and practice are also discussed. However, the research has three main questions with corresponding sub-questions. These RQs have been studied and answered as follows:

RQ1. What software development methodology and application lifecycle should be applied? Agile software development methodology, in combination with the objectives and key results framework have been applied for developing this research project [6-8]. Indeed, Agile and OKRs approaches have been used throughout the software system design and development phases.

RQ2. Can QR Codes be utilised as a transport tool for financial data? During the research, it was found that QR codes can be used as a medium for transferring financial information between two devices, as demonstrated in FTM Sequence steps shown in Table 1 and the derived algorithms discussed in Section 4. The QR codes in FTM steps are only used to pass secure information between two devices, and therefore, they can be checked by the QRBee system (RQ2.1). For security and ethical reasons, the data in QR code needs to be secured; otherwise, such private data, as users' information and credit card data, will be exposed to any individual who scans the QR code. Encryption and security check processes were implemented to prevent such issues (RQ2.2). Table 1 has the FTM sequence 8 steps describe and states what data is being transferred at each step of the process (RQ2.3).

RQ3. What are the key techniques to ensure the integrity of secure financial transactions? The research encryption has digital signature verification processes, which used the RSA asymmetric cryptography algorithm for producing digital signatures and encryption for protecting sensitive Client data. The digital signature allows the QRBee system to verify data integrity as well as the user identity. This technique prevents situations when a third party tries to manipulate data to its own needs (RQ3.1). The data itself is encrypted, meaning that it cannot be read without the knowledge of private RSA keys to prevent various malicious events from happening (RQ3.2).

The QRBee framework ideas came from prior research studies, specifically drawing from [4, 11-14] study sources. For instance, the QRBee infrastructure aligns seamlessly with the established mobile payment security framework as described by Ahmed et al. [2]. Indeed, the primary emphasis of the QRBee system centres on enhancing transaction security and user convenience. There are several software development tools have been used, including Microsoft Visual Studio, Xamarin, and C# programming language. This offers an exceptional rapid development framework, complemented by a comprehensive set of programming tools, such as a robust compiler, debugger, profiling tools, source version control, and a package manager. This combination enables quick software system prototyping and accommodates various software development methods. In this research project, Agile methodology was utilised for the software development process and control.

The selection of the real-time database for QRBee system was driven by a focus on performance. While traditional SQL databases like Oracle or Microsoft SQL Server meet performance criteria, they come with associated costs [14]. In contrast, modern NoSQL cloud realtime databases offer high performance, more flexibility, and frequently come with contemporary programming UIs at almost no cost. QRBee system utilises MongoDB, which is available under a free community license. However, the trade-off with these databases lies in their distinctive interfaces, deviating from the standardised relational database model commonly known as SQL database systems. In addition, the QRBee system algorithms in Section 4 describe the processes and data flow to ensure the security of the FTM. These algorithms find applicability in a category of communication protocols suitable for implementing payment systems, or to some extent, any secure data exchange protocols that do not necessitate internet connectivity on the Client side. An exemplary application of these algorithms could be observed in the secure stamping of client-provided data. In such a scenario, the Client is armed with a smartphone and engages in a secure exchange with the Merchant situated at a standalone kiosk or computer without internet access.

The proposed QRBee FTM protocol ensures both transaction security and the confidentiality of personal information (data privacy), making it applicable across non-secure channels such as the internet and QR codes exchange. Although QRBee system utilises QR codes in the initial phase of the financial transaction, it is important to note that the FTM protocol itself is not confined to QR codes. One can envision scenarios where the same information can be transmitted via alternative channels, such as NFC or Bluetooth, revealing the FTM protocol's adaptability beyond QR code usage. Also, the QRBee system establishes a robust groundwork and foundation for a real-world practical commercial software application. The inherent characteristics of QR codes impose limits on the security robustness of the FTM protocol. Given that any RSA-encrypted value shares the length of the RSA key used for encryption or digital signature calculation, the size of the generated QR code is inevitably influenced. Both the mobile-phone's screen and camera have limitations on the size of the QR code for effective recognition. Consequently, the use of long keys is practically restricted. Although this is not currently an issue, it could pose a challenge in the future as specific key lengths may be deemed insecure.

Further, it is essential to acknowledge that the survey findings rely on the subjective opinions of the participants, and the participant pool was relatively small. Consequently, the survey results only reflect the broader public sentiment regarding QR Codes and their utilisation in financial transaction software applications. The distinctive feature of the FTM protocol, requiring only Merchants to have internet access for the transaction process, sets QRBee framework apart from other payment approaches. This unique characteristic enables its application in service points with limited or no cellular reception. In such scenarios, Merchants can utilise private broadband connections and secure Wi-Fi endpoints without the need to establish public Wi-Fi for Clients. This not only allows Merchants to save costs on providing public Wi-Fi, but also, eliminates the requirement for Know Your Client (KYC) procedures on public networks (Client identification) and ensures compliance with data protection regulations (e.g. GDPR 2018). For Clients, this translates to not enabling Wi-Fi connections on their phones or connecting to insecure public Wi-Fi. In essence, this expedites the payment process, which enhances Client turnaround, and therefore, boosts Merchant revenue.

Furthermore, the QRBee system satisfied the research purpose, as it was able to handle financial transactions with the use of QR codes, but there are areas for further development, as listed in Table 2. This includes various tasks, which can be a base for further research studies. For instance, a research study can be used to delve deeper into the security aspect of the proposed QRBee framework and find a way to integrate it into different payment systems. The QRBee framework presents several notable tasks and considerations for enhancement (Table 2). Firstly, the absence of a card payment system gateway impedes the execution of real card transactions. Moreover, the web application, tailored for back-office support, lacks features for front-end operations but addresses essential functions like KYC procedures, reporting, and security measures, such as fraud detection and client notifications. Indeed, the QRBee developed application is exclusive to Android platforms. To fortify the overall system, there is a call for robust security measures encompassing both the mobile and potential web applications. Lastly, compliance with legal requirements mandates the incorporation of KYC procedures into the registration process, particularly for new Clients and Merchants.

Table 2: Further development.

№	Task	Description
1	Payment gateway	There is no card payment system gateway implemented.
2	Web application	Not part of the project, the web-based tool may support back-office operations and KYC procedures, reporting and security routines such as fraud detection, client notifications, and malicious merchant blocking.
3	iOS Client application	QRBee application was developed for Android platforms only, but iOS support was intended.
4	Security measures	Security measures need to be implemented for a potential web application.
5	KYC support	New Clients and especially Merchants' registration should incorporate KYC procedures.

Conclusion

This research study introduces the QRBee framework, and a mobile software application, which has been designed for financial transactions using an API network server, QR code technology, and real-time cloud database server. The QRBee framework comprises a meticulously tested mobile application that operate on Android phones and a MongoDB instance to facilitate seamless secure payment model. The monetary transaction protocol has been discussed, which relies on QR code exchange and presents a rapid and secure method for conducting mobile payments. Indeed, the protocol is hardware-agnostic and can be seamlessly implemented using smartphones. The payment process uniquely demands only Merchants to have an internet connection, which offers a cost-saving by eliminating the need of public internet-network access in areas with limited or no network connectivity. The protocol's security is ensured through the utilisation of digital signatures and asymmetric cryptography encryption algorithms. Finally, the implementation process underwent detailed examination and analysis, which revealed test results that affirm the capability of the implemented transaction method to transfer sensitive information for financial transactions through communication networks. The software architecture and development processes adhered to the Agile methodology through Scrum practice. The resulting prototype Android mobile application is designed to be cloud-ready with the ability to be executed within a Docker container on the cloud network using QRBee framework for secure financial transactions.

Acknowledgements

None

Conflict of Interest

The authors declare that there is no conflict of interest.

References

1. KWABENA GY, MEI Q, GHUMRO TH, LI W, ERUSALKINA D (2021) Effects of a Technological-Organizational-Environmental Factor on the Adoption of the Mobile Payment System. J Asian Financ Econ Bus 8: 329-338.
2. Ahmed W, Rasool A, Javed AR, Kumar N, Gadekallu TR, et al. (2021) Security in Next Generation Mobile Payment Systems: A Comprehensive Survey. IEEE Access 9: 115932-115950.
3. Yong A, Rana ME, Shanmugam K (2022) Improved Shopping Experience Through RFID Based Smart Shopping System. Int Conf Decis Aid Sci Appl IEEE p. 635-644.
4. Nseir S, Hirzallah N, Aqel M (2013) A secure mobile payment system using QR code. 2013 5th Int. Conf. Comput Sci Inf Technol IEEE p. 111-114.
5. Soon TJ (2008) Information Technology Standard Committee. Sect. three — QR Code, Singapore: iTSC p.59-78.
6. Biesialska K, Franch X, Muntés-Mulero V (2021) Big Data analytics in Agile software development: A systematic mapping study. IST 132: 106448.
7. Stray V, Moe NB, Vedal H, Berntzen M (2022) Using Objectives and Key Results (OKRs) and Slack: A Case Study of Coordination in Large-Scale Distributed Agile. Proc. 55th Hawaii Int. Conf. Syst. Sci.
8. González Moyano C, Pufahl L, Weber I, Mendling J (2022) Uses of business process modeling in agile software development projects. IST 152: 107028.
9. Schwaber K (2025) Professional Scrum Framework. Scrum.Org.
10. Troelsen A, Japikse P (2022) Pro C# 10 with .NET 6. 11st ed. Berkeley, CA: Apress.
11. Chang V, Chen W, Xu QA, Xiong C (2021) Towards the Customers' Intention to Use QR Codes in Mobile Payments. J Glob Inf Manag 29: 1-21.
12. Ahmad L, Al-Sabha R, Al-Haj A (2021) Design and Implementation of a Secure QR Payment System Based on Visual Cryptography. 7th ICIM: 40-44.
13. Sintyaningrum DE, Muladi, Ashar M (2021) The Encryption of Electronic Professional Certificate by Using Digital Signature and QR Code. 2021 Int. Conf. Converging Technol. Electr Inf Eng, IEEE p. 19-24.
14. Ohyver M, Moniaga J V, Sungkawa I, Subagyo BE, Chandra IA (2019) The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test. Procedia Comput Sci 157: 396-405.
15. Kaur K, Sachdeva M (2017) Performance evaluation of NewSQL databases. Int Conf Inven Syst Control, IEEE p. 1-5.
16. Pendse S, Krishnaswamy V, Kulkarni K, Li Y, Lahiri T, et al. (2020) Oracle Database In-Memory on Active Data Guard: Real-time Analytics on a Standby Database. IEEE 36th Int Conf Data Eng, IEEE p. 1570-1578.
17. AbouGrad H, Chakhar S, Abubahia A (2023) Decision Making by Applying Machine Learning Techniques to Mitigate Spam SMS Attacks. Key Digital Trends in Artificial Intelligence and Robotics 670: 154-166.
18. AbouGrad H, Shabarshov A (2024) Mobile Payment System & FinTech to Process Financial Transactions.
19. Alliance Software (2025). What are Software Development Methodologies? Alliance Software.
20. Doerr J, Page L (2018) Measure What Matters: How Google, Bono, and the Gates Foundation Rock the World with OKRs. Penguin Publishing Group.
21. Martin RC (2020) Clean Agile: Back to Basics. 1st editio. Boston: Pearson.
22. Alsaqqa S, Sawalha S, Abdel-Nabi H (2020) Agile Software Development: Methodologies and Trends. Int J Interact Mob Technol 14(11): 246.
23. AbouGrada H, Qadoosa A, Sankurua L (2025) Financial Decision-Making AI-Framework to Predict Stock Price Using LSTM Algorithm and NLP-Driven Sentiment Analysis Model.
24. Schuldt H (2017) Multi-tier Architecture. In: Liu L, Özsu MT, editors. Encycl Database Syst, New York, NY: Springer New York p. 1-3.
25. Ayob NZB, Hussin ARC, Dahlan HM (2009) Three Layers Design Guideline for Mobile Application. Int Conf Inf Manag Eng, IEEE: p. 427-431.
26. Bradshaw S, Brazil E, Chodorow K (2019) MongoDB: The Definitive Guide, 3rd Edition. 3rd editio. O'Reilly Media, Inc.
27. Baek W, Minh CC, Trautmann M, Kozyrakis C, Olukotun K (2007) The OpenTM Transactional Application Programming Interface. 16th Int Conf Parallel Archit Compil Tech (PACT 2007) IEEE: p.376-387.
28. Boicea A, Radulescu F, Agapin LI (2012) MongoDB vs Oracle -- Database Comparison. 2012 Third Int Conf Emerg Intell Data Web Technol, IEEE: p.330-335.
29. Alese B, Falaki E (2012) Comparative Analysis of Public-Key Encryption Schemes. Int J Eng Technol 2(9): 1552-1568.
30. Liddy C, Sturgeon A (1999) The evolution of certificate model architecture. Inf Manag Comput Secur 7(2): 95–100.
31. Divyanjali, Ankur, Pareek V. (2014) An Overview of Cryptographically Secure Pseudorandom Number Generators and BBS. Int J Comput Appl; ICACEA: 19-28.
32. Bernstein PA, Newcomer E (2009) Principles of Transaction Processing. 2nd ed. Elsevier Science.
33. Eren BA (2022) QR code m-payment from a customer experience perspective. J Financ Serv Mark 29: 106-121.
34. Wang S, Yao X, Gong D, Tu H (2023) Overlapping community detection in software ecosystem based on pheromone guided personalized PageRank algorithm. Inf Softw Technol 163: 107283.