

P.G.D.A.V. College (M), University Of Delhi



B. Sc. (H) Computer Science (II Year)

SEMESTER IV

Software Engineering Project

SUPERVISOR:

Dr. Aparna Datt

SUBMITTED BY:

Anwasha

Sanyal(3906)(20053570041)

Kartik

Joshi(3932)(20053570016)

SECURE SCAN

CERTIFICATE

This is to certify that the project entitled, “SECURE SCAN” has been done by: **ANWESHA SANYAL** and **KARTIK JOSHI** of Bachelor of Science in Computer Science during semester IV from P.G.D.A.V.(M) College ,University of Delhi under the supervision of **Dr. APARNA DATT**.

DECLARATION

I hereby declare that this Project Report titled “SECURE SCAN,” submitted to the Department of Computer Science, PGDAV College (Delhi University), is a record of original work done by the team under the guidance of Dr. Aparna Datt.

The information and data given in the report are authentic to the best of the team's knowledge. This Project Report is not submitted to any other university or institution for awarding any degree, diploma, or fellowship or published at any time before.

Acknowledgement

The contentment achieved on the successful completion of any task is incomplete without mentioning the names of the people who made it possible with their consistent guidance, support and necessary encouragement. The Project was jointly undertaken by Anwesha Sanyal and Kartik Joshi as their 4th semester Software Engineering Project, under the able guidance and supervision of Dr Aparna Datt. Our primary thanks to her, who poured over every inch of our Project with proper attention and helped us throughout the working of the Project. It's our privilege to acknowledge our most profound gratitude to her for her inspiration which has helped us immensely. We are incredibly grateful to her for her unstilted support and encouragement in preparing this Project.

INDEX

→ Problem Statement

- (I) Problem definition
- (II) Solution to the problem

→ Process Model

- (I) V model
- (II) Requirements

→ Software Requirements Specifications(SRS)

- (I) SRS
- (II) Product Functions
- (IV) Modules/Functionalities

→ Use Case

→ Data Flow Diagram

- (I) Level 0 DFD
- (II) Level 1 DFD

→ Entity Relationship Diagram

→ RDBMS Tables

→ Relational Database Executed in SQL

→ Timeline

→ User Interface

→ Pseudocode

- (I) Verify User Module

→ Flow Chart

- (I) Verify User Module

→ Control Flow Graph(CFG)

- (I) Verify User Module

→ Cyclomatic Complexity

- (I) Verify User Module

→ Testing

- (I) Black Box Testing
- (II) White Box Testing

→ Test Cases

- (I) Verify User Module

INTRODUCTION TO THE PROBLEM STATEMENT:

OVERVIEW

One-step way to make payments, know about an item, and much more work in one scan. The technology of the QR code has facilitated us. A QR code, an initialism for quick response code, is a type of matrix barcode or two-dimensional barcode that has become a part of our daily life in numerous ways. When we scan a QR code, it merely displays the link we can follow. However, there always exists a question mark on the URL source, whether that link is secured enough.

EXISTING SYSTEM:

There are numerous applications available for scanning QR Codes. As the QR codes are not inspected or made by a secure authority, anyone can create a QR code using simple techniques in a matter of seconds, yet this piece of code is incomprehensible to the naked eye.

The URL data type is the only scenario in which conventional QR codes can carry executable data. Because a reader would generally send the data to the application associated with the data type utilized by the QR code, these URLs may contain JavaScript code that can be used to exploit vulnerabilities in applications on the host system, such as the reader, web browser, or image viewer.

Linking to dangerous websites with browser exploits, enabling the microphone/camera/GPS and then streaming those feeds to a remote server, analysis of sensitive data (passwords, files, contacts, transactions), sending email/SMS/IM messages or packets for DDoS as part of a botnet, corrupting privacy settings, stealing identity, and even containing malicious logic such as JavaScript or a virus are all risks. These actions could occur in the background, with the user only seeing the reader open a seemingly innocuous web page.

OUR SYSTEM:

Even though QR code scanner is inseparable from our daily life, many users are unaware of the dark sides of using this technology. One way is to check each and every QR code and its source and authenticity. Scanning all the QR codes around a large area, maintaining a proper database, and going through it each time while scanning a code is a laborious task. An easier way would be to automate this task and make it more efficient and feasible by maintaining a middleware database authenticator software. That software will work as a standard QR code scanner and, on scanning, verifies the code from the database and informs the user whether the code is secure or not.

Software Processes are a set of actions that are used to specify, develop, implement, and test software systems. A software process model is an abstract representation of a process that provides a description of the process from a certain point of view.

The software model that fits our project is the “**V-model**” (a variation of the Water-fall Model). V- model provides a way of visualizing how verification and validation actions are applied to early engineering works. As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once the code has been generated, the team moves up the right side of the V, performing a series of tests that validate each of the models created as the team moves down the left side.



2. Requirement analysis and specification:

- **Requirement gathering and analysis:**

- **Requirement specification:**

3. Design:

- #### 4. Coding and Unit Testing:

- During the coding phase, the software design is converted into source code using any programming language that is appropriate. As a result, every designed module is coded. The goal of the unit testing phase is to see if each module is functioning properly.

5. Integration and System Testing:

- The integration of various modules occurs shortly after they have been coded and unit tested. The integration of various modules is done in stages over a period of time. Previously planned modules are added to the partially integrated system during each integration step, and the resultant system is tested. Finally, after all of the modules have been successfully integrated and tested, a complete working system is obtained, and system testing is performed.

There are three types of testing activities in system testing, as described below:

- **Alpha testing:** Alpha testing is the system testing performed by the development team.
- **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
- **Acceptance testing:** After the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it.

6. Maintenance:

- The most crucial phase of a software life cycle is maintenance. Maintenance takes up 60% of the total time and effort required to develop a complete software. Maintenance can be divided into three categories:
 - **Corrective Maintenance:** This type of maintenance is performed to correct errors that were not discovered during the development phase of the product.
 - **Perfective Maintenance:** This type of maintenance is performed in response to a customer's request to improve the system's functionality.
 - **Adaptive Maintenance:** When porting software to a new environment, such as working on a new computer platform or with a new operating system, adaptive maintenance is usually required.

Why use a V-model?

The Secure Scanner Project is very simple and is easy to understand. We are giving extensions to already existing applications like normal QR Code Scanners.

- V-Model is used where requirements are well understood and changes can be required if error occurs during the testing phase in our project.
- Processes, actions and results are documented but needs verification that can be done only at the time of testing.
- This project reinforces good habits: define-before-design and design-before code. This project will be built using Android-Studio IDE and Java AWT. It can be further upgraded to IOS.
- The scope of our project precludes a purely linear process. V-Model is best suitable for linear processes.

Functional Requirements:

Scanning Module:

- This is the main module of the application that a user views just after opening the app. It provides a GUI where the user is accessible to scan the QR code and view the result of the query.

Verified User Registration Module:

- This module will only be for users who want them to be verified. Using this module, users can place a request to access the features available for verified users.

Login Module:

- This module will only be accessible by verified users. Using this module, users can enter as a verified user.

Request Module:

- This module will only be accessible by verified users. Using this module, users can request a query claiming an URL to be unsafe.

Design and Implementation Constraints:

- This is an Android Based Application using Java which can be further extended to IOS also.
- This project is built using Android-Studio IDE and Java AWT.
- The application uses a MYSQL database server.

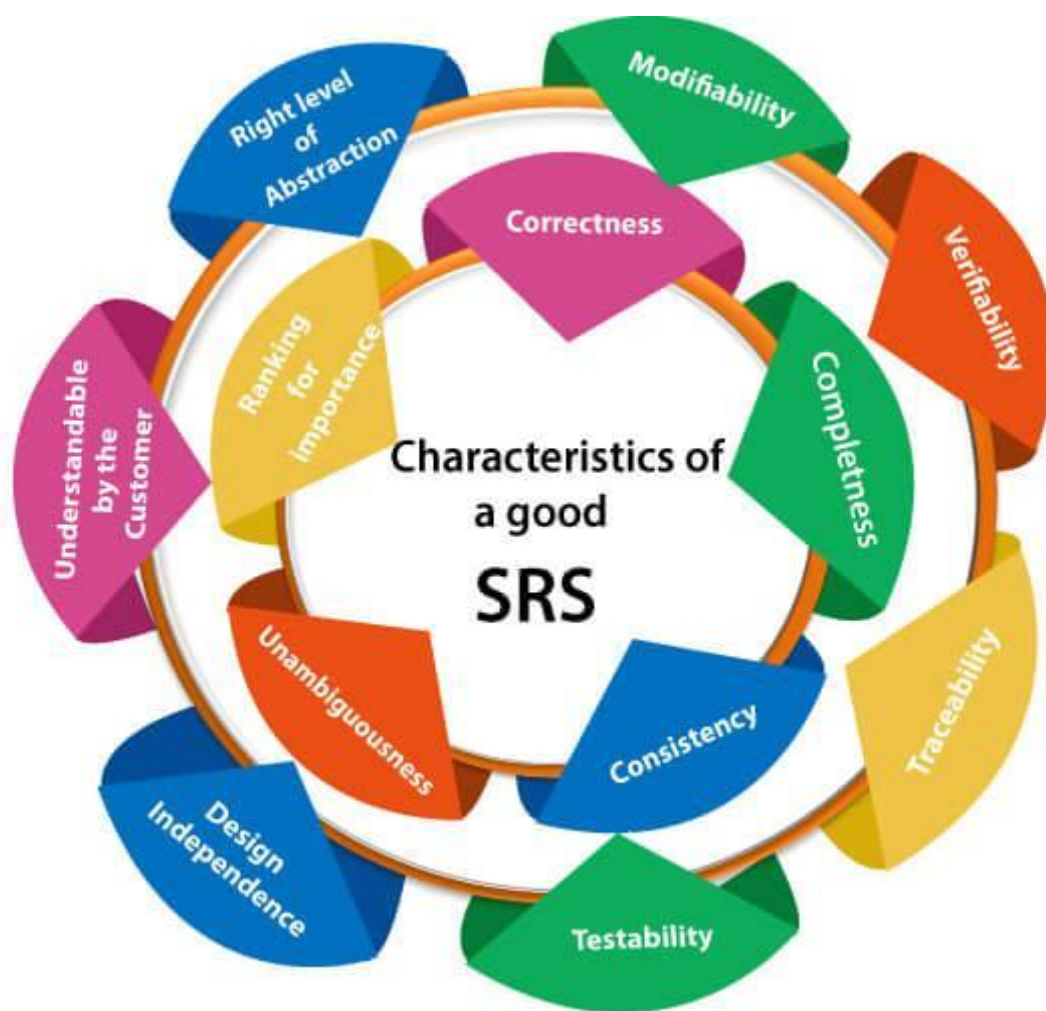
Software Requirement Specification (SRS):**What is SRS?**

A software requirements specification is a document that captures complete description about how the system is expected to perform.

SRS Overview:

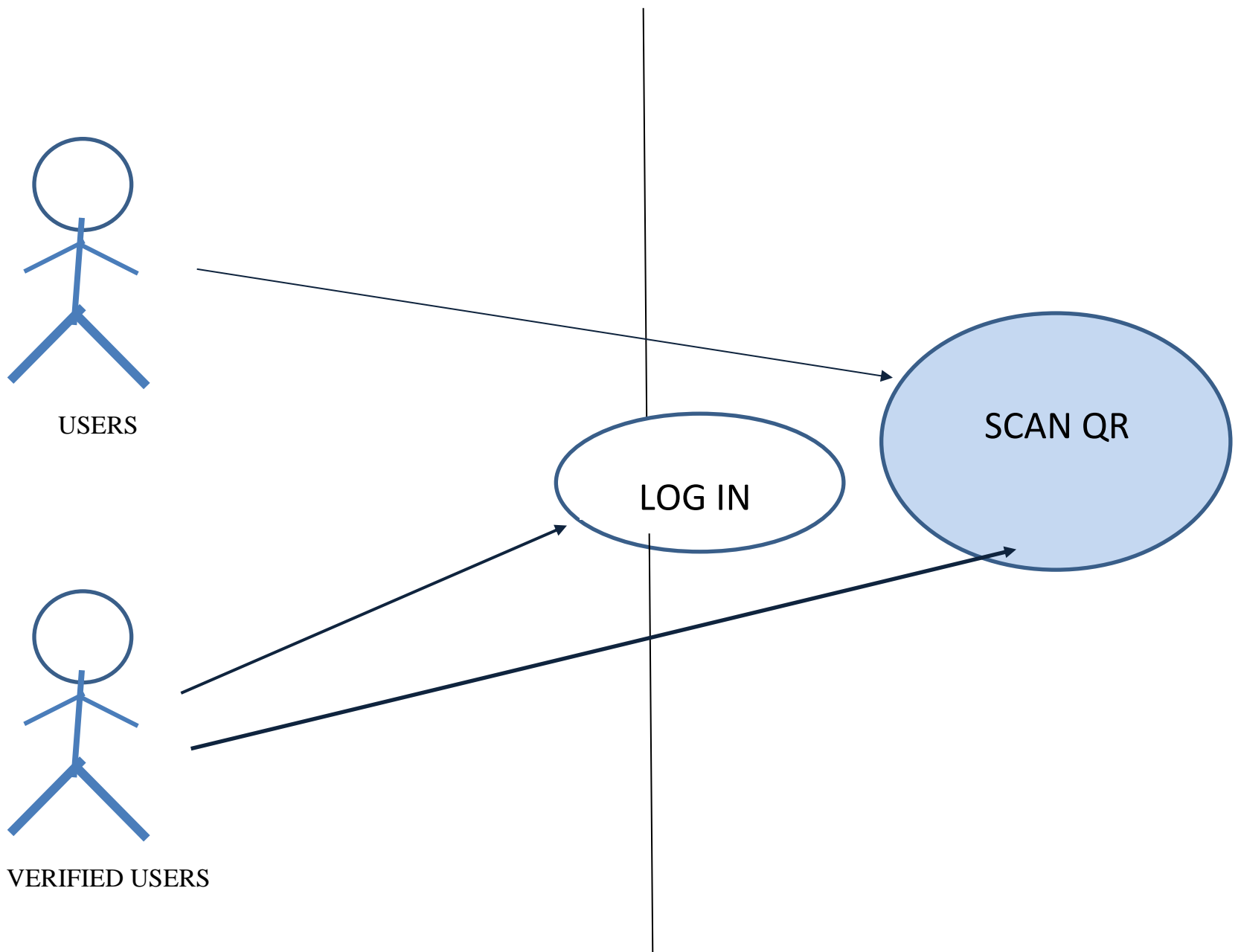
The SRS document satisfies the following: -

1. It specifies the external system behavior.
2. It specifies constraints on the implementation.
3. It is easy to change.
4. It serves as a reference tool for system maintainers.
5. It records forethought about the lifecycle of the system.
6. It characterizes acceptable responses to undesired events.

**Product Functions:**

- This application aims to provide users security while scanning QR codes so that users can save themselves from the scams.
- Users will need to scan a QR code that'll be redirected to the associated link.
- Using our app users will get a security score of that link.
- Output screen will show a score out of 10 that'll define the security of the website.
- One can create his/her profile and that will be verified by the moderators. The user has to submit documents to get validated.
- Once the validation is completed, verified users can put up a request to enlist a link as unsafe.

User Diagram:



Scanning:

Introduction: This use case describes how a user scans a QR Code.

Actors: Users

Pre-Conditions: None

Post-Conditions: A security score for the link associated with the QR code will be displayed and after that user can move forward.

Basic Flow: This use case initiates its function when an actor wants to use the application to scan a QR Code.

- Using a normal phone camera, the user scans the QR Code s/he wants to.
- According to the link's authenticity, the app displays a security score for the QR link s/he wants to visit.

Alternative Flow: None

Special Requirements: None

Use Case Relationships: COMMUNICATES

Verified User Registration:

Introduction: This use case describes how a user can get upgraded to a verified user(Moderator).

Actors: Users

Pre-Conditions: None

Post-Conditions: If the use case is successful, the actor is upgraded to a verified user. If not, the system state is unchanged.

Basic Flow: This use case initiates its function when the actor wants to put a request for marking an QR code as unsafe.

- Application requests the actor to submit his/her relevant documents.
- Application validates the documents and if those are found to be correct then it allows the actor to be logged in as a verified user.

Alternative Flows:

Invalid documents: If due to some reason the basic flow gets interrupted, the actor gets 2 options. First, s/he can go back to the basic page for normal users. Second, can opt for another request to get his/her request verified.

Special Requirements: None

Use Case Relationships: EXTENDED

Log in:

Introduction: This use case describes how a verified user logs into the QR Code scanner app.

Actors: Verified Users

Pre-Conditions: Relevant documents must be authenticated.

Post-Conditions: If the use case is successful, the actor is logged into the system. If not, the system state is unchanged.

Basic Flow: This use case initiates its function when the actor wants to login to the application.

- Application requests the actor to enter his/her username and password.
- Actor enters his/her name and password.
- Application validates the inputs and if those are found to be correct then it allows the actor to be logged in.

Alternative Flows:

Invalid Username or Password: If due to some reason the basic flow gets interrupted, the actor gets 2 options. First, s/he can go back to the basic page for normal users. Second, you can opt for a forgotten password/username.

Special Requirements: None

Use Case Relationships: INCLUDES

Request:

Introduction: This use case describes how a verified user requests for a QR Code to be unsafe.

Actors: Verified Users

Pre-Conditions: None

Post-Conditions: If the use case is successful, the request will be approved. If not, the system state is unchanged.

Basic Flow: This use case initiates its function when the actor wants to put up a request for a QR code.

- Application requests the actor to input the QR code, reason to mark it as unsafe, proof of the QR to be vulnerable.
- Actor fills the required input fields.
- Application validates the inputs and if those are found to be correct then it approves the request and make the changes in the database.

Alternative Flows:

Invalid Request: If due to some reason the basic flow gets interrupted, the actor gets 2 options. First, s/he will receive that the request is invalid. Second, s/he can opt for another request.

Special Requirements: None

Use Case Relationships: INCLUDES

Architectural design

Requirements of the software should be transformed into an architecture that describes the software's top-level structure and identifies its components. This is accomplished through architectural design (also called system design), which acts as a preliminary 'blueprint' from which software can be developed. IEEE defines architectural design as 'the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.' This framework is established by examining the software requirements document and designing a model for providing implementation details. These details are used to specify the components of the system along with their inputs, outputs, functions, and the interaction between them. An architectural design performs the following functions.

It defines an abstraction level at which the designers can specify the functional and performance behaviour of the system.

- 1. It acts as a guideline for enhancing the system (whenever required) by describing those features of the system that can be modified easily without affecting the system integrity.
- 2. It evaluates all top-level designs.
- 3. It develops and documents top-level design for the external and internal interfaces.
- 4. It develops preliminary versions of user documentation.
- 5. It defines and documents preliminary test requirements and the schedule for software integration.

Though the architectural design is the responsibility of developers, some other people like user representatives, systems engineers, hardware engineers, and operations personnel are also involved. All these stakeholders must also be consulted while reviewing the architectural design in order to minimize the risks and errors.

Architectural Design Representation

Architectural design can be represented using the following models.

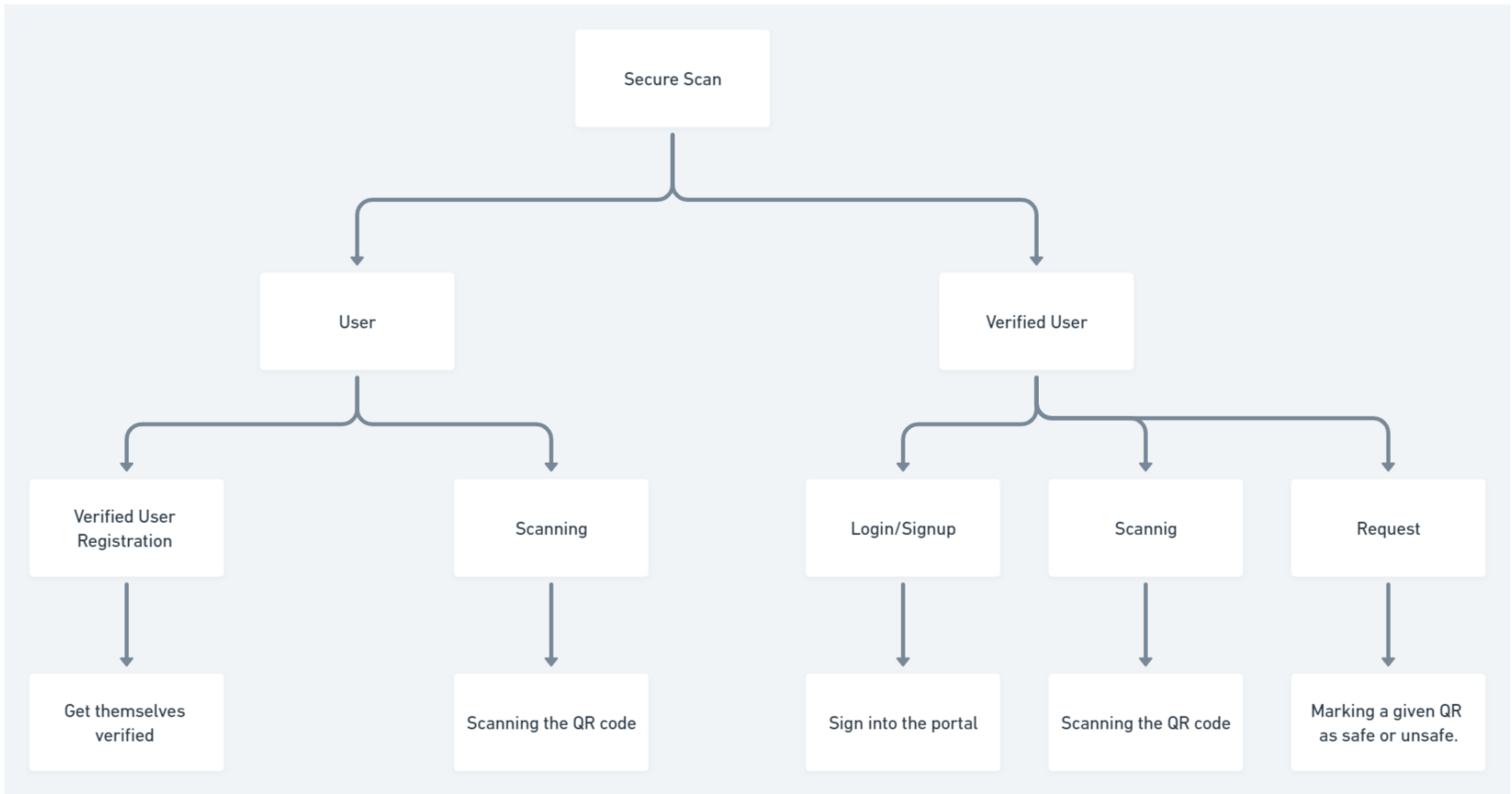
Structural model: Illustrates architecture as an ordered collection of program components

Dynamic model: Specifies the behavioral aspect of the software architecture and indicates how the structure or system configuration changes as the function changes due to change in the external environment.

Process model: Focuses on the design of the business or technical process, which must be implemented in the system

Functional model: Represents the functional hierarchy of a system

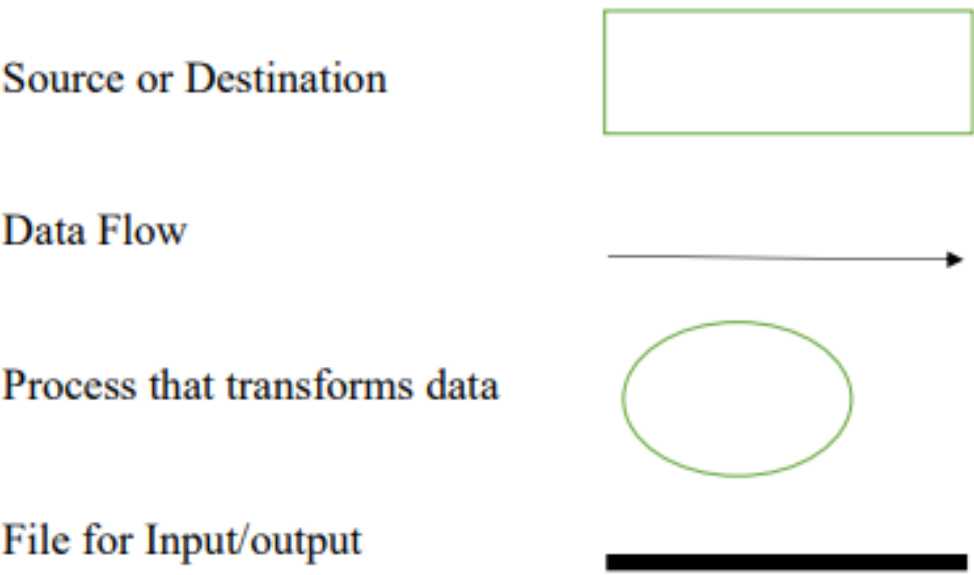
Framework model: Attempts to identify repeatable architectural design patterns encountered in similar types of application. This leads to an increase in the level of abstraction.



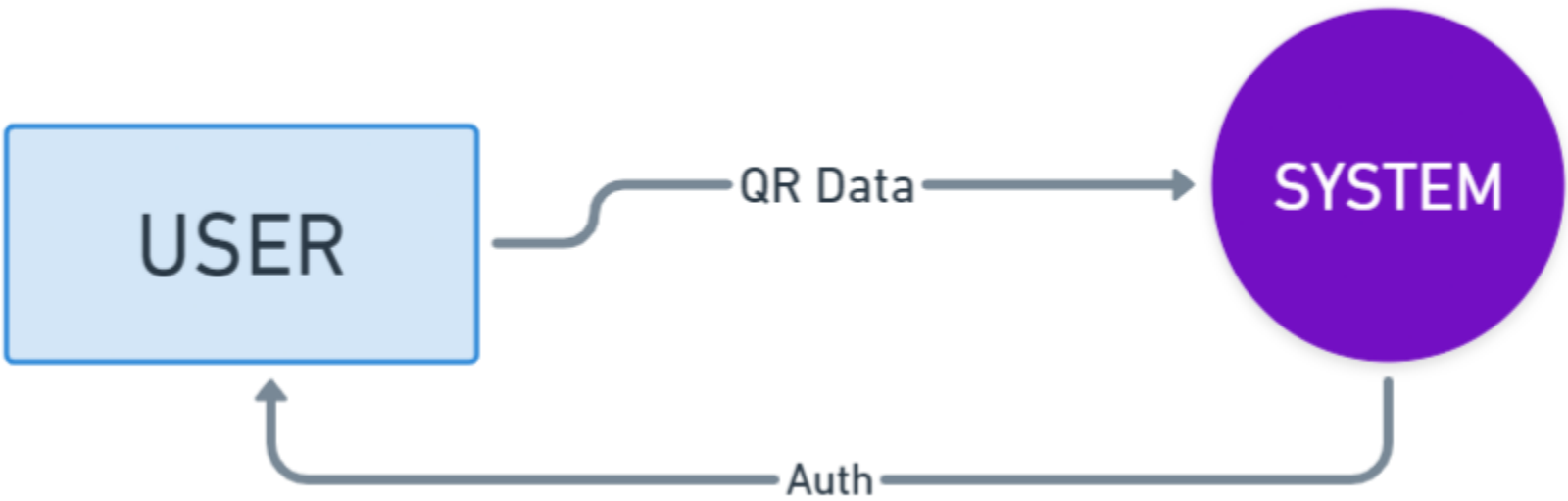
Data Flow Diagram:

A Data Flow Diagram is a graphical representation of the "flow" of data through an information system, which models the process characteristics. A DFD is frequently used as a basic step in the development of a system overview that can later be refined.

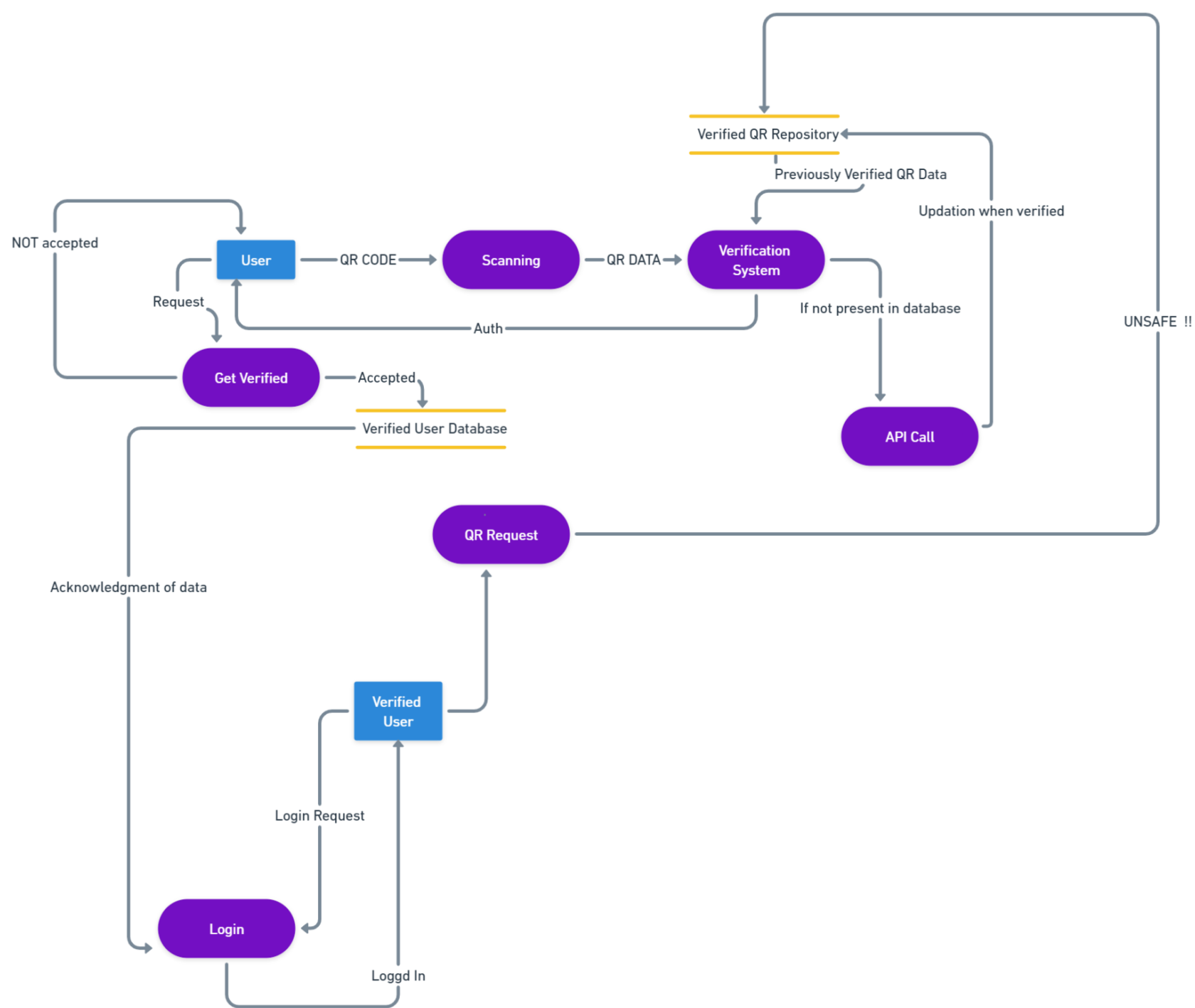
- DFD can also be used for the visualization of data processing.
- A data flow diagram (DFD) depicts the types of data that will be input to and output from the system, as well as where the data will come from and go to, and where it will be kept. It doesn't show information about process timing or whether processes will run in order or in parallel.



Level 0 Data Flow Diagram:



Level 1 Data Flow Diagram:



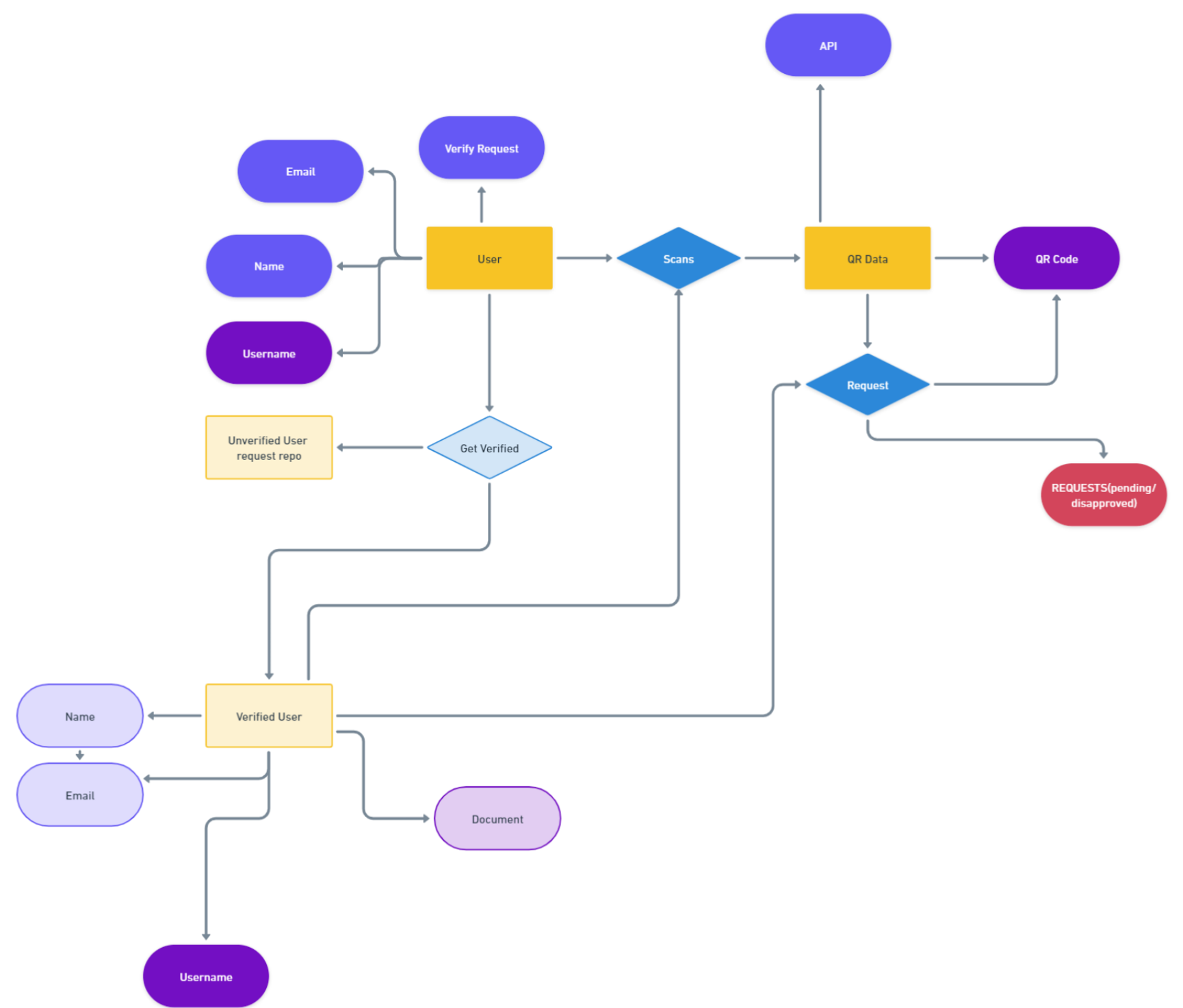
Design of Database

Tables The data to be used in the system are stored in various tables. The number of tables used and their structure are decided upon keeping in mind the logical relation in the data available.

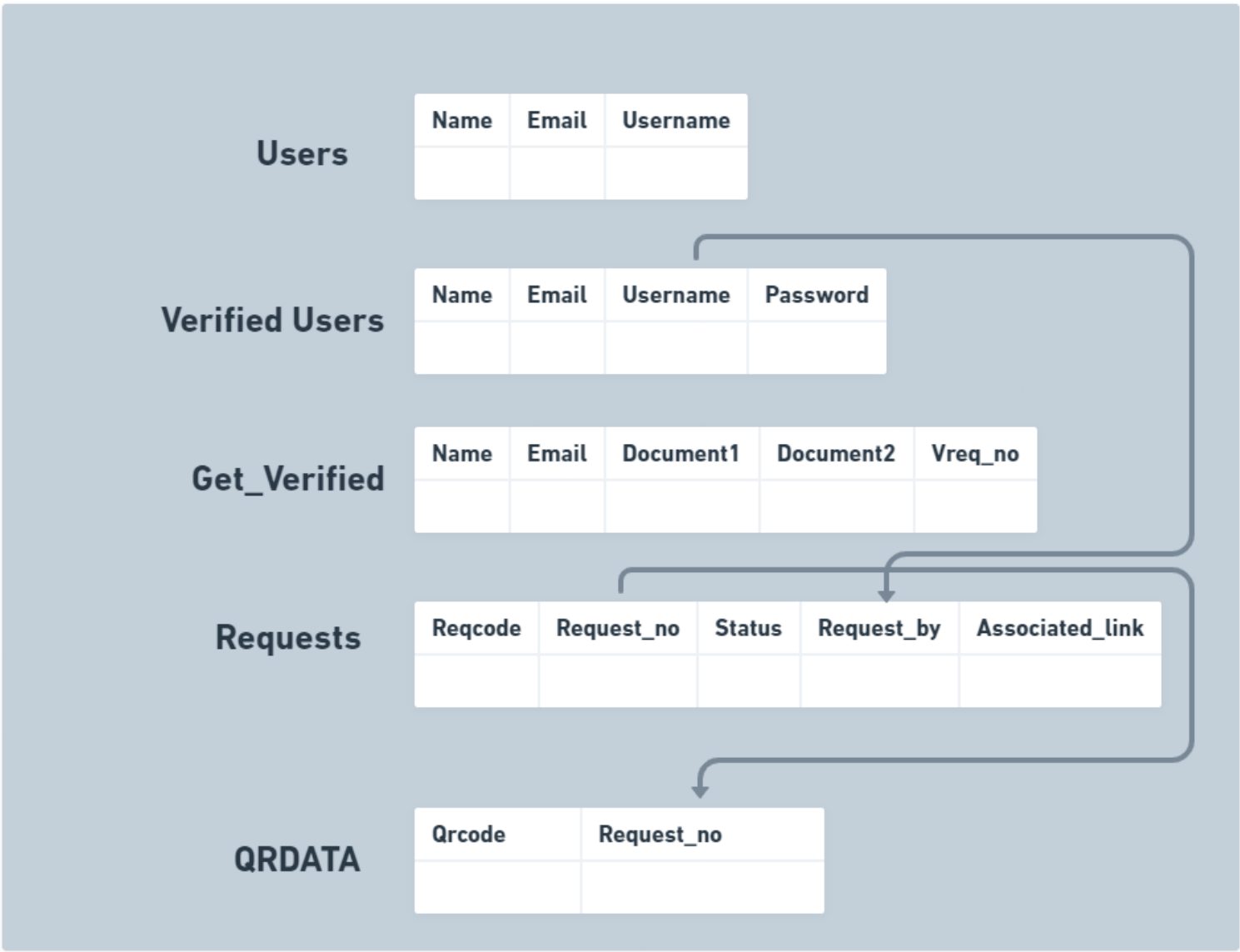
The database design specifies:

1. The various tables to be used.
2. Data to store in each table.
3. Format of the fields and their types.

ERD(Entity Relationship Diagram)



RDBMS:



Database Creation Using SQL:

Tables in Secure_Scan:

```
mysql> show tables;
+-----+
| Tables_in_secure_scan |
+-----+
| get_verified          |
| qrdata                |
| requests              |
| users                 |
| verified_users        |
+-----+
5 rows in set (0.01 sec)
```

Users’ Table:

```
mysql> desc users;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| NAME       | varchar(30)   | NO   |     | NULL    |       |
| USERNAME   | varchar(30)   | NO   | PRI | NULL    |       |
| EMAIL      | varchar(50)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```


Verified Users’ Table:

```
mysql> desc verified_users;
```

Field	Type	Null	Key	Default	Extra
NAME	varchar(30)	NO		NULL	
USERNAME	varchar(30)	NO	PRI	NULL	
EMAIL	varchar(50)	NO		NULL	
PASSWORD	varchar(20)	NO		NULL	

4 rows in set (0.01 sec)

Table for Normal user to be Verified:

```
mysql> desc get_verified;
```

Field	Type	Null	Key	Default	Extra
NAME	varchar(30)	NO		NULL	
EMAIL	varchar(50)	NO		NULL	
DOCUMENT1	varchar(64)	NO		NULL	
DOCUMENT2	varchar(64)	NO		NULL	
VREQ_NO	int	NO	PRI	NULL	

5 rows in set (0.00 sec)

Table for all QR_Code:

```
mysql> desc qrdata;
```

Field	Type	Null	Key	Default	Extra
QRCODE	varchar(64)	NO	PRI	NULL	
REQUEST_NO	int	YES	MUL	NULL	

2 rows in set (0.00 sec)

Table for storing the QRs requested by the verified users:

```
mysql> desc requests;
```

Field	Type	Null	Key	Default	Extra
REQCODE	varchar(64)	NO		NULL	
REQUEST_NO	int	NO	PRI	NULL	
STATUS	int	NO		NULL	
REQUEST_BY	varchar(30)	YES	MUL	NULL	
ASSOCIATED_LINK	varchar(500)	YES		NULL	

5 rows in set (0.00 sec)

Gantt Chart

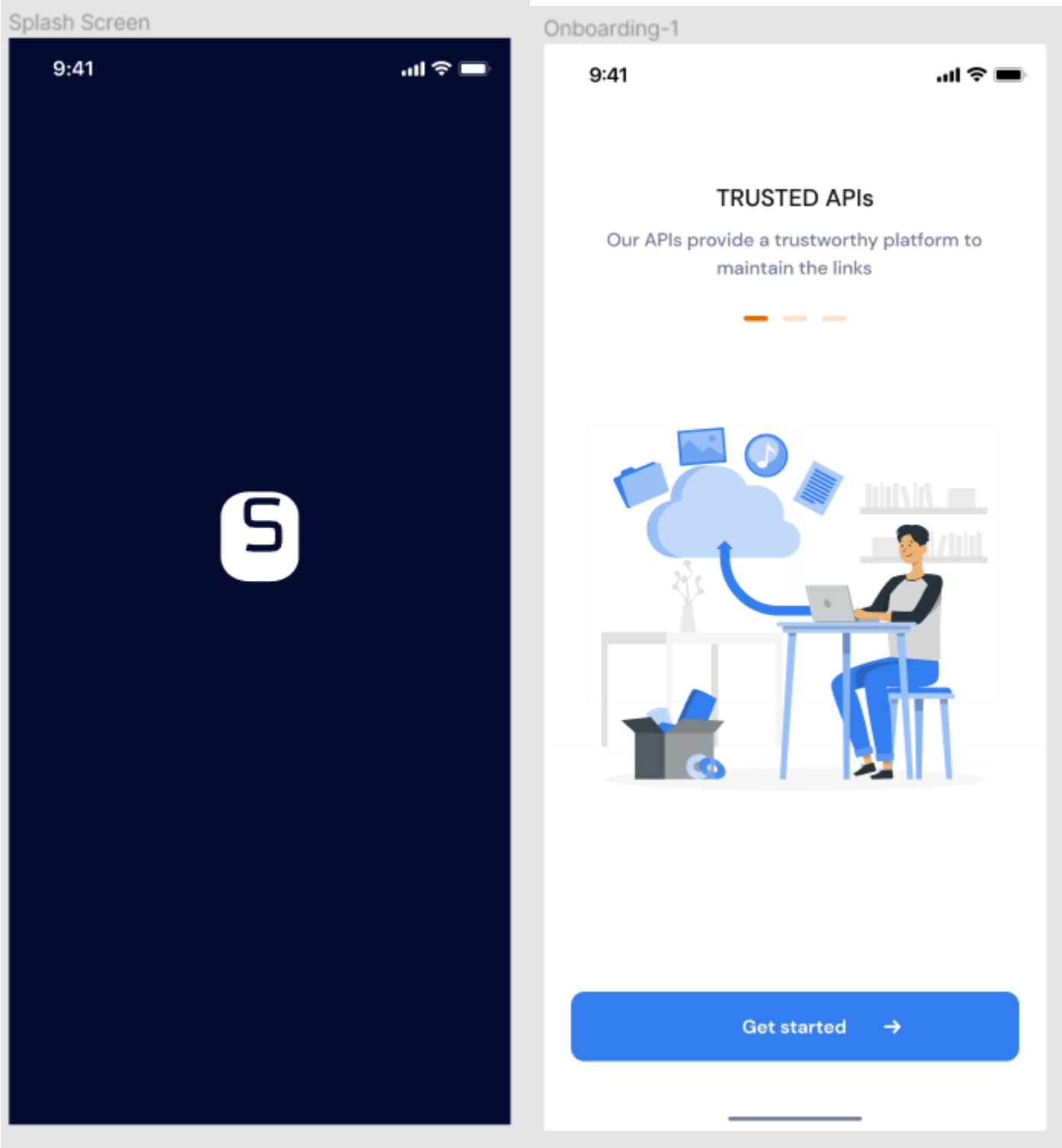
When creating a software project schedule, you begin with a set of tasks (the work breakdown structure). If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals. As a consequence of this input, a time-line chart, also called a Gantt chart, is generated. A time-line chart can be developed for the entire project. Alternatively, separate charts can be developed for each project function or for each individual working on the project. The following figure illustrates the format of a time-line chart. It depicts a part of a software project schedule that emphasizes the concept scoping task for a word processing (WP) software product. All project tasks (for concept scoping) are listed in the left-hand column. The horizontal bars indicate the duration of each task. When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones.

SE PROJECT

[illegible]

GUI(Graphic User Interface):

Starting screen of the application.



9:41



GET SECURITY INDEX CHECKED

Know about the links and get it's security
Index



Get started →

Sign up

9:41

CREATE AN ACCOUNT

Input the right details to set up your account the right way.

Email address

Phone number

Password


Get started →

Already have an account? Sign In

Verification

9:41

←



Verification link has been sent to your email

Open email app

Didn't get the verification code?

Resend code

9:41



VERIFICATION CODE

We've sent a verification code to your email, please type it in the field below

Enter verification code

Continue →

Already have an account? [Sign In](#)

9:41



SIGN IN TO ACCOUNT

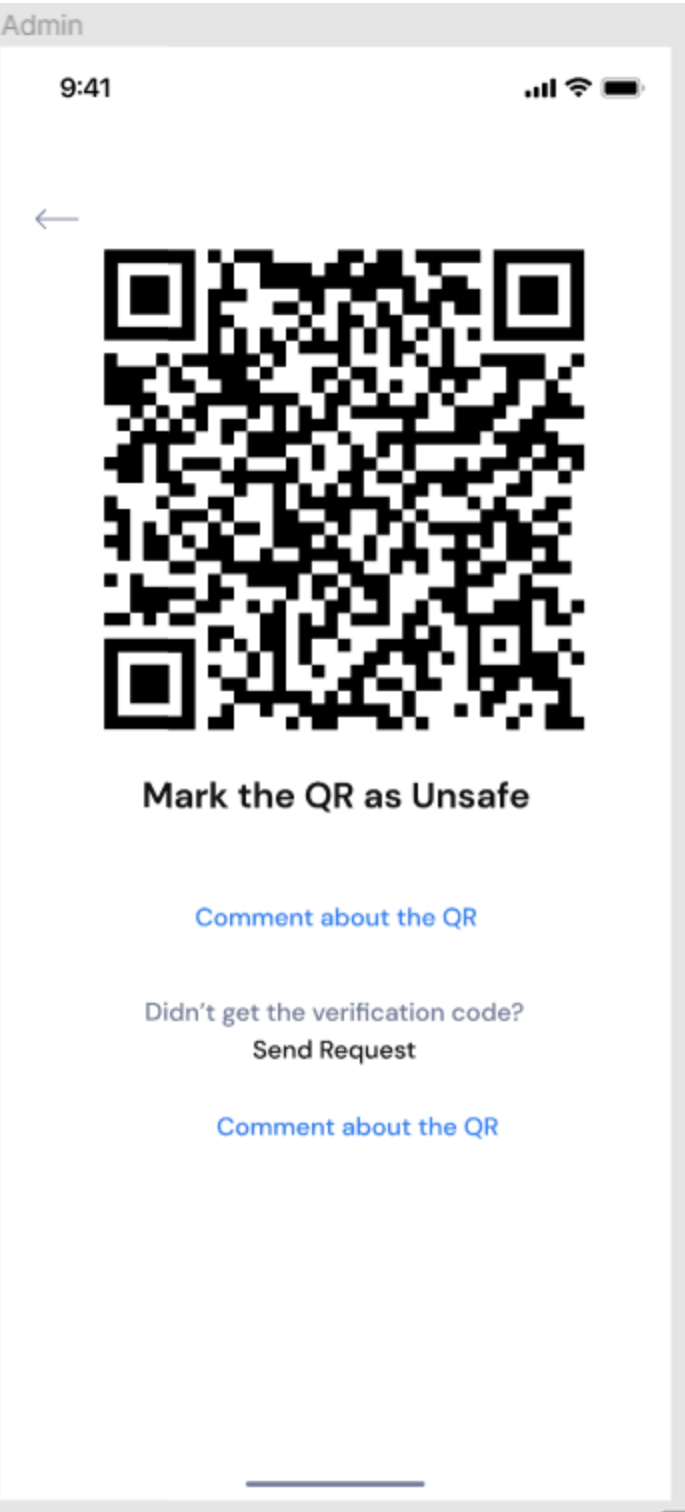
Input the right details to login in into your account the right way.

Username

Password

Log In →

Want to get verified? [Sign Up](#)



9:41

Hello, Welcome

Recent Scans

Files

QR6.png

April 19, 2020

300kb

QR5.png

April 19, 2020

300kb

QR4.png

April 19, 2020

300kb

QR3.png

April 19, 2020

300kb

QR2.png

April 19, 2020

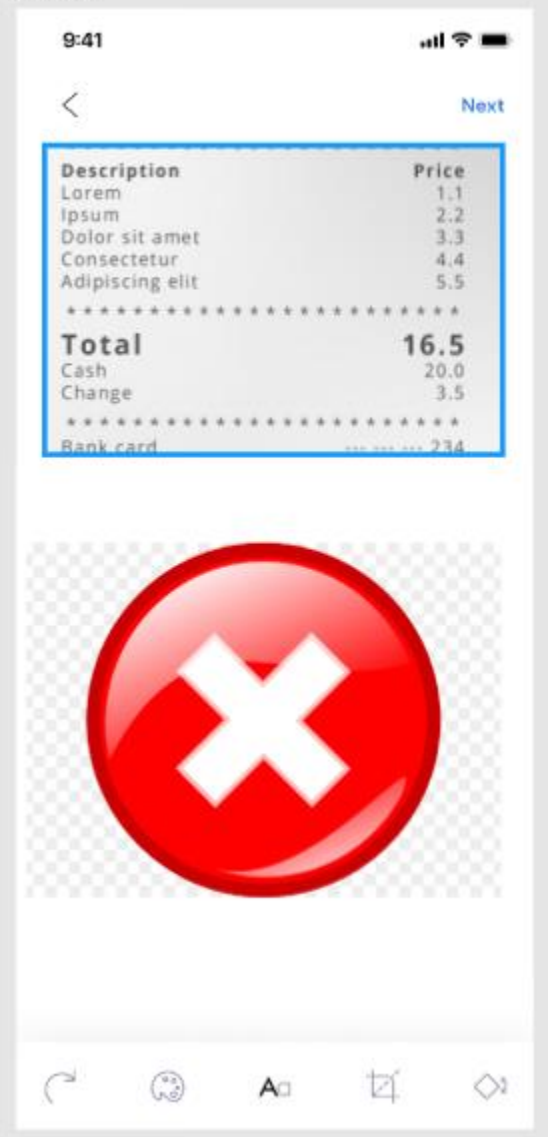
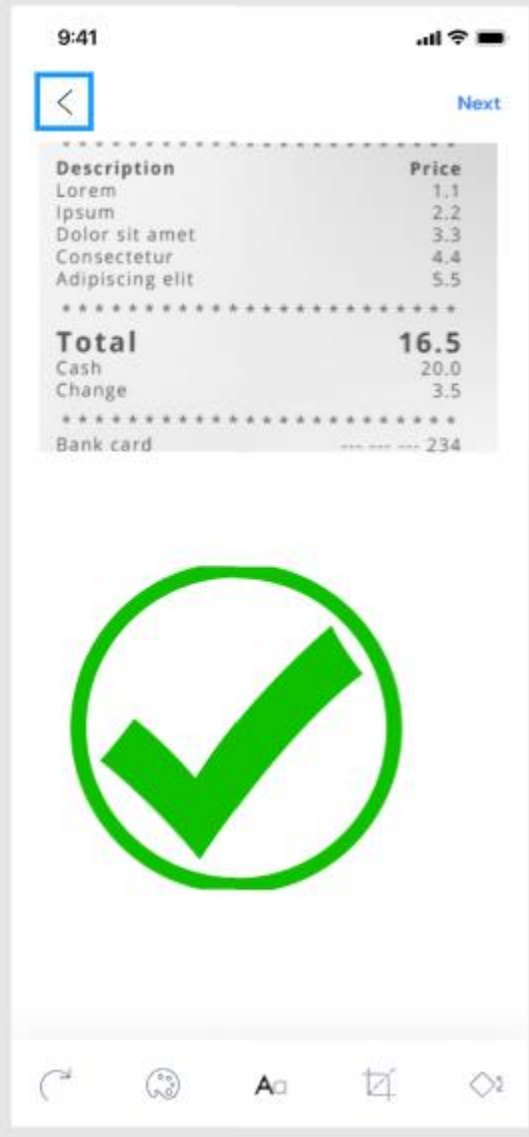
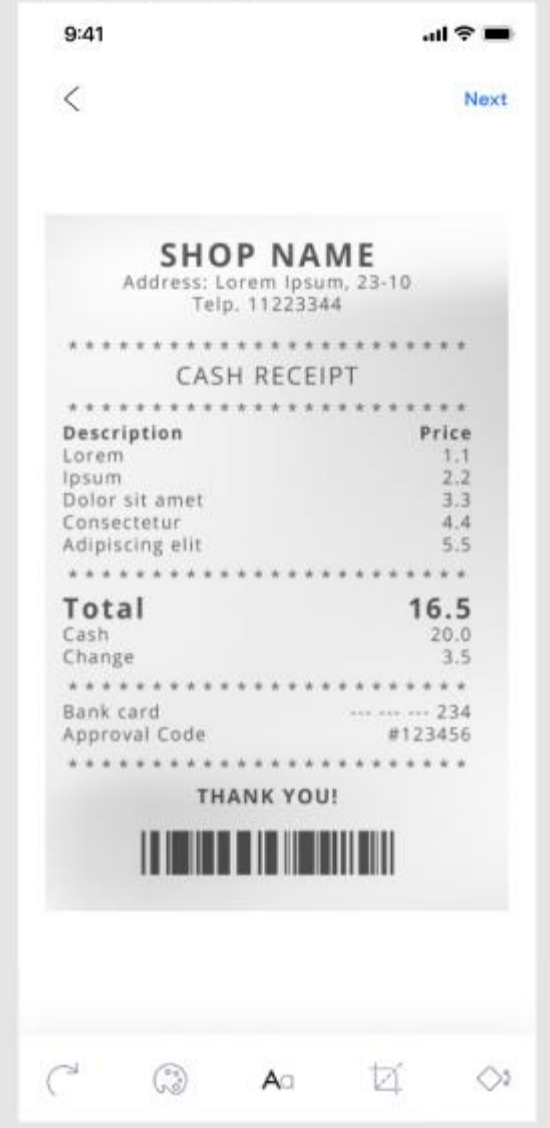
300kb

QR1.png

April 19, 2020

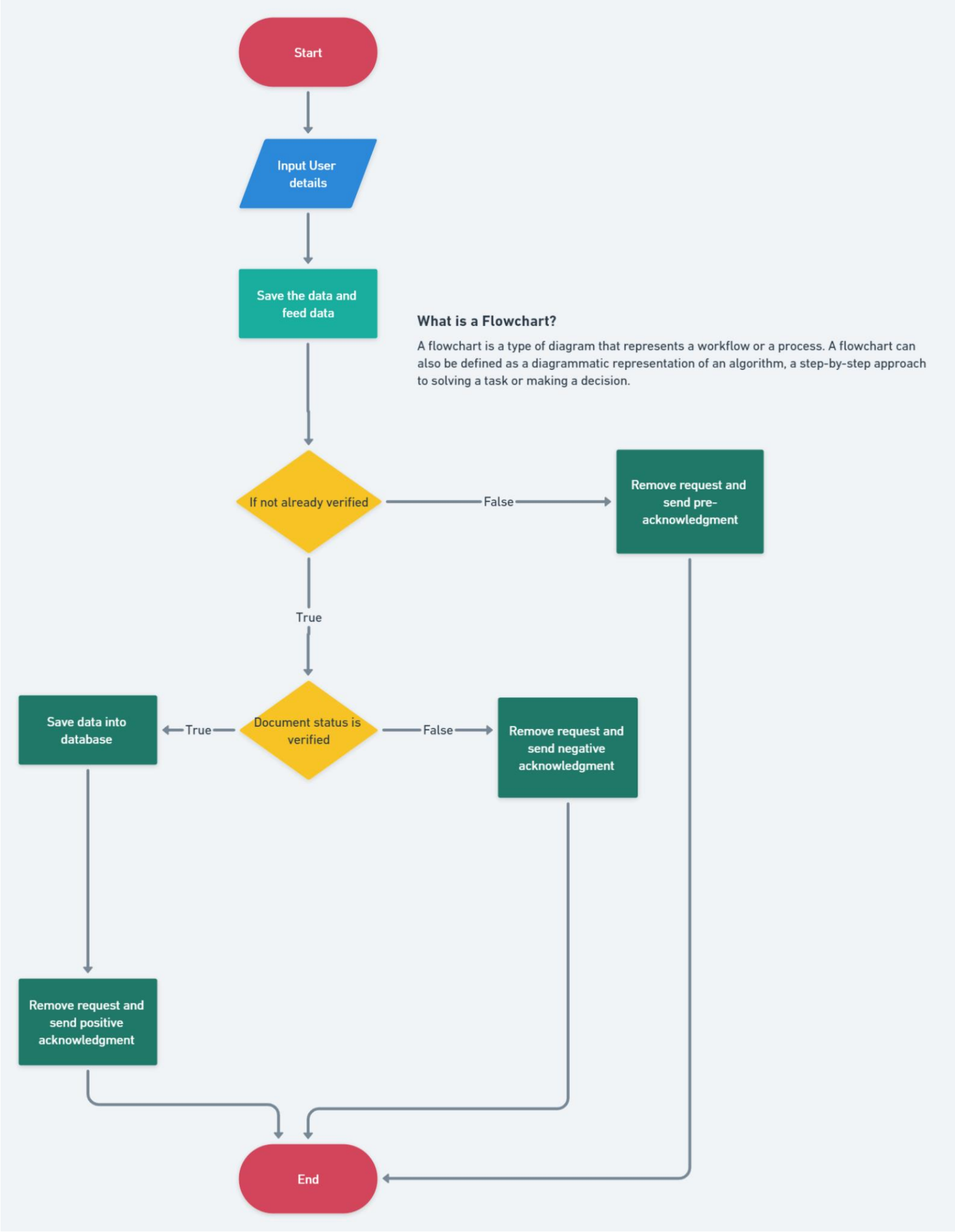
300kb

Quick actions



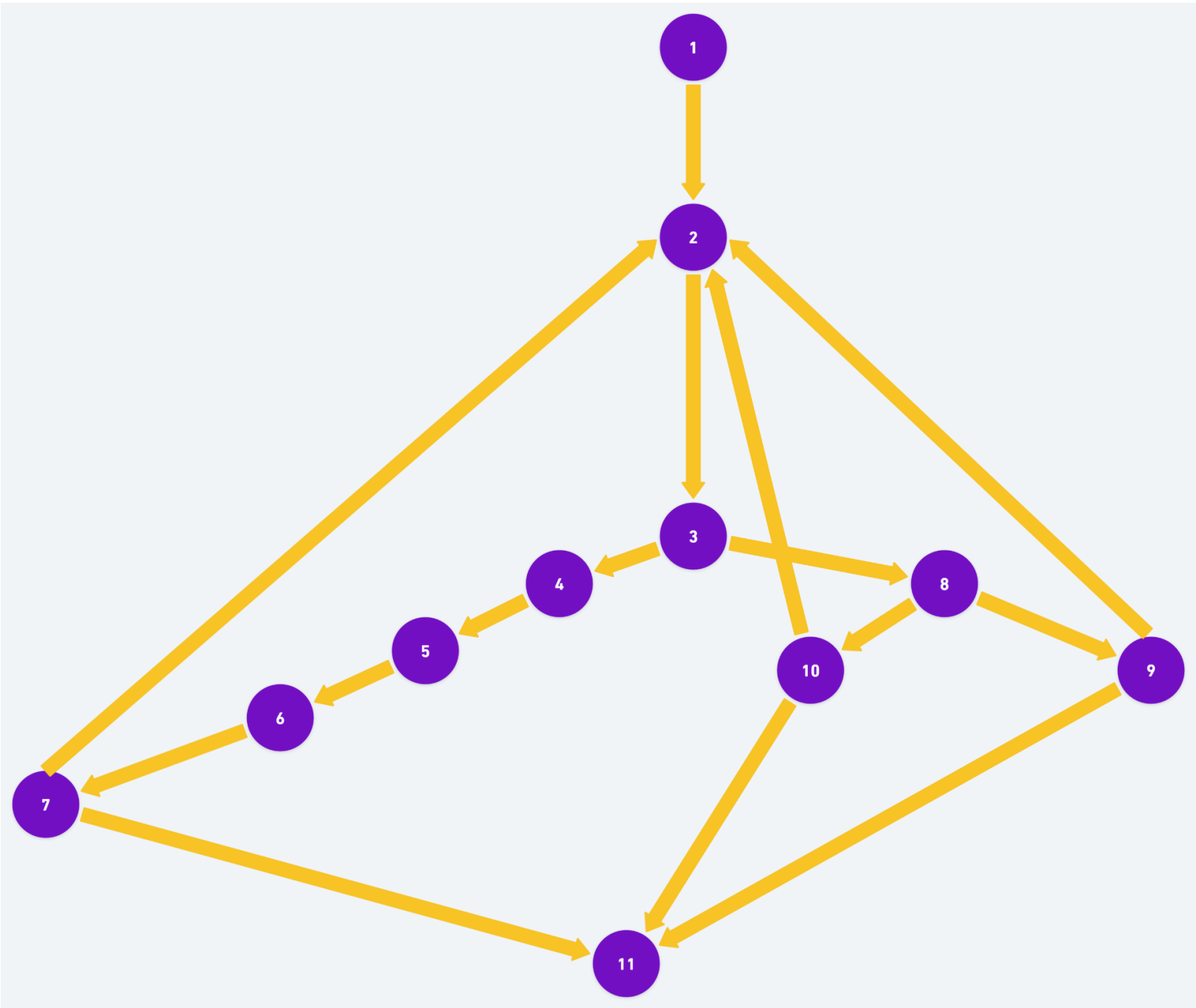
Flow Chart

A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes, and arrows to demonstrate a process or a program. With algorithms, we can easily understand a program. The main purpose of using a flowchart is to analyze different methods.



Control Flow Graph

A Control Flow Graph (CFG) is the graphical representation of control flow or computation during the execution of programs or applications. Control flow graphs are mostly used in static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit.



Cyclomatic Complexity

Cyclomatic complexity is a software metric used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. It is computed using the control-flow graph of the program: the nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command. Cyclomatic complexity may also be applied to individual functions, modules, methods or classes within a program.

Method1:

Cyclomatic Complexity = Number of Edges - Number of Nodes +2

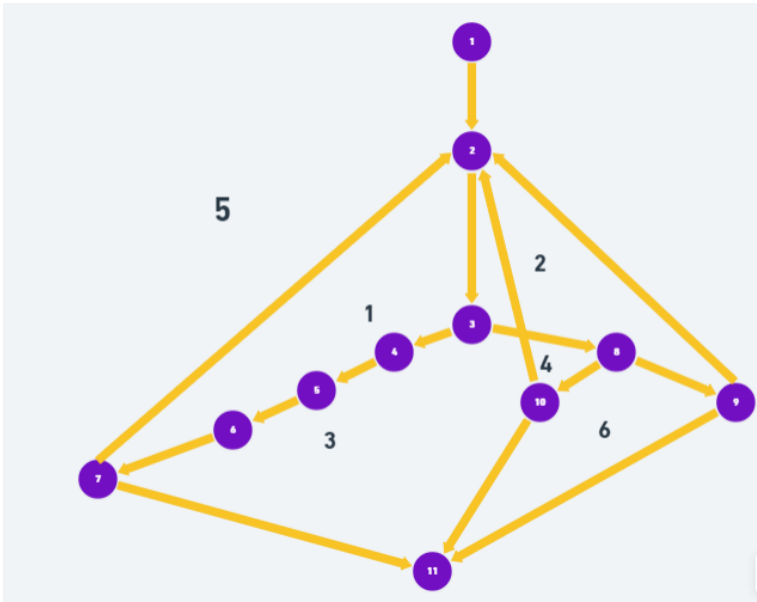
$$=15-11+2= 6$$

Method2:

Cyclomatic Complexity=Π+1 (number of predicate nodes+1)

$$=5+1=6$$

Method3:



Cyclomatic Complexity=No. of Circular Region

=6

Pseudo Code

Pseudocode is a plain language description of the steps in an algorithm or another system. Pseudocode often uses structural conventions of a normal programming language, but is intended for human reading rather than machine reading. It typically omits details that are essential for machine understanding of the algorithm, such as variable declarations and language-specific code.

Module Explained:

Get Verified

code:

```
{
1.
class user = new user;
input user.name =user.get(name);
input user.email =user.get(email);
input user.doc1 = user.get(doc1);
input user.doc2 = user.get(doc2);
input user.doc3 = user.get(doc3);
2.
save(user);

for(auto per_request:user.req)
{
3.if(!per_request.docverified && verify(per_requeest.doc))
{
4. per_request.docverified = true;
```

```

5.save verifiedUser = (user.per_request);

6.delete(user.per_request);

7.acknowledgment_Req(user.email);

}

else

{

8.if(per_request.docVerified)

9.notify(per_request.email);

else

10.rejected_Req(per_request.email);

}

}

}

```

TESTING

- Testing is the process of executing a program with the intent of finding an error. A good test case has a high probability of finding - undiscovered error.
- Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, coding.
- The purpose of product testing is to verify and validate the various work products viz. units, integrated unit, final product to ensure that they meet their requirements.

1. Acceptance testing:

Acceptance testing is used when the software is developed for a specific customer. A series of tests are conducted to enable the customer to validate all requirements. These tests are conducted by the end user/customer and may range from ad hoc tests to well-planned systematic series of tests.

2. Alpha testing:

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developer's site. Alpha testing is often employed for off-the -shelf software as a form of internal acceptance testing, before the software goes to beta testing.

3. Beta testing:

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Black Box Testing:

- Black Box testing is also called functional testing.
- Black Box Testing is a test case design method that focuses on the functional requirements of the software that enables the software engineer to derive a set of input conditions that fully exercise all functional requirements for a program.

- Test the artifacts from the external point of view.
- Specifications are used to test data that is what type of input should be given to the unit or module should be specified.
- We can check the functionality on the basis of the output generated and the input, not looking at the internal coding.
- It attempts to find errors in the following categories
 1. Incorrect or missing functions
 2. Interface errors
 3. Errors in data structure or External database access
 4. Behavior or performance error
 5. Initialisation and termination errors

White Box Testing:

- It is also called glass box testing.
- White Box testing is a test case design method that uses the control structure of the procedural design to derive test cases.
- Using White Box Testing method, the software engineer can derive test cases that
 1. Guarantee that all independent paths within a module have been exercised at least once.
 2. Exercise all logical decisions on their true and false sides.
 3. Execute all loops at their boundaries and within their operational bounds.
 4. Exercise internal data structures ensure their validity.
- Test the artifacts from the internal point of view.
- It cannot detect absence of features.
- For security purposes the Email of the user is required in case he/she forgets his/her password and wants to retrieve that.

Test Cases:

Sl No.	Test Cases	Expected Output
1.	Document verified and user not verified	added to verified user list
2.	Document verified and user verified	notify user that user is already verified
3.	Document not verified and user also not verified	reject user’s requests
4.	Document not verified and user verified	notify user that user is already verified

FUNCTION POINTS

Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since ‘functionality’ cannot be measured directly, it must be derived indirectly using other direct measures. Function points are derived using an empirical relationship based on countable (direct) measures of software’s information domain and assessment of software complexity. Information domain values are defined in the following manner:

- **Number of user inputs:** Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.
- **Number of user outputs:** Each user output that provides application oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
- **Number of user inquiries:** An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an online output. Each distinct inquiry is counted.
- **Number of files:** Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.
- **Number of external interfaces:** All machine readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.

Once these data have been collected, the following table is completed and a complexity value is associated with each count.

- **External Inputs :**

1. Login Credentials : Username,Password
2. Admin Login - Username, Password
3. User requests to be verified: Name, Email, doc1, doc2,doc3
- **External Output :**

1. Scan Result - Safety Score
2. User to be verified message
- **Logical Internal Files :**

1. QR data
2. Verified User Details
3. Get Verified Requests
- **External Interface Files**
1. API Calls
- **External inquiries: -**

External Inputs = 9
External Outputs = 2
Internal Logical Files = 3
External Inquiries = 0
External Interfaces = 1

Function units	Low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Table : Unadjusted Functional Point

Information Domain Values	Count	Weighing Factor - Simple	Weighing Count
External Inputs	1	3	3
External Outputs	2	4	8
External Inquiries	1	3	3
Internal Logical Files	3	7	21

External Logical Files	0	5	0
Count Total			35

Table : Function Point Table

S.NO.	Questions	VAF
1.	Data Communication	4
2.	Distributed Data Processing	4
3.	Performance	5
4.	Heavily Used Configuration	0
5.	Transaction Role	2
6.	Online Data Entry	5
7.	End-User Efficiency	0
8.	Online Update	5
9.	Complex Processing	5
10.	Reusability	5
11.	Installation Ease	4
12.	Operational Ease	5
13.	Multiple Sites	0
14.	Facilitate Change	5

Table : Project Estimators

Degree	Significance
0	Not Present
1	Insignificant Influence
2	Moderate Influence
3	Average Influence
4	Significant Influence
5	Strong Influence

Total Degree of Influence (TDI) = $\sum(f_i) = 49$

The f_i (i = 1 to 14) are “*Complexity Adjustment Values*” based on responses.

Computing Function Points :

$$FP = Count_total * (0.65 + 0.01 * \sum(f_i))$$

where,count total is the sum of all FP entries

$$FP = 35 * (0.65 + 0.01 * 49) \\ = 39.9$$

RISK ANALYSIS

A table provides a project manager with a simple technique for risk production. A risk table is sorted by probability and impact to rank risks. A project team begins by listing all risks in the 1st column of the table. This can be accomplished with the help of the risk item checklist referenced. Each risk is categorized in the 2nd column. The probability of occurrence of each risk is entered in the next column of the table. Next, the impact of each risk is assessed. Each risk component is assessed using the characterization presented and an impact category is determined. The categories for each of the four risk components-performance, support, cost and schedule-are averaged to determine an overall impact value. Once the first four columns of the risk table have been completed, the table is sorted by probability and by impact. High-probability, high-impact, risk-impact risks percolate to the top of the table and low-probability risks drop to the bottom.

S.No	Risk	Risk Category	Prob. Of Occurrence	Weight Factor Or Loss	RE(Risk Exposure) or Risk Factor	RMMM(Risk mitigation,monitoring,and management)
1.	Hardware Crash of the server side system containing the QR repo	PREDICTABLE RISK	10%	4	0.4	Maintaining a backup cloud repo of the QR DATA and updating it timely.
2.	End user comfortability with the software usage	BUSINESS RISK	20%	3	0.6	Have good communication during process cycles.
3.	Error in the updation of wrong QR in the repo	TECHNICAL RISK	10%	3	0.3	Re-verify the QR before adding to the repo.
4.	API failure	TECHNICAL RISK	20%	3	0.6	Use of higher quality of bandwidth to avoid traffic
5.	If some team member leave the project in between the process	TECHNICAL RISK	30%	2	0.6	Have a backup team of skilled people
6.	Loss of the user login data	TECHNICAL RISK	10%	5	0.5	Uses of secure and safer databases managment

BIBLIOGRAPHY

- Software Engineering – A Practitioner’s Approach o By Roger S. Pressman
- Software Engineering- K.K. Aggarwaland Yogesh Singh
- en.wikipedia.org