

# **Project Report**

1. **INTRODUCTION**
  - 1.1 Project Overview
  - 1.2 Purpose
2. **LITERATURE SURVEY**
  - 2.1 Existing problem
  - 2.2 References
  - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
  - 3.1 Empathy Map Canvas
  - 3.2 Ideation & Brainstorming
  - 3.3 Proposed Solution
  - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
  - 4.1 Functional requirement
  - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
  - 5.1 Data Flow Diagrams
  - 5.2 Solution & Technical Architecture
  - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
  - 6.1 Sprint Planning & Estimation
  - 6.2 Sprint Delivery Schedule
  - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
  - 7.1 Feature 1
  - 7.2 Feature 2
  - 7.3 Database Schema (if Applicable)
8. **TESTING**
  - 8.1 Test Cases
  - 8.2 User Acceptance Testing
9. **RESULTS**
  - 9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
  - Source Code
  - GitHub & Project Demo Link

# 1. INTRODUCTION

## 1.1 Project Overview

Artificial intelligence (AI) has been a subject of interest in the research field for the past few years. It has now been brought closer to commercial use due to recent technological advances and speedier data accessibility. Its relevance to global business models is underlined by the significant investments in it made by Internet powerhouses including Google, YouTube, Amazon and Facebook. In the banking sector where data is of substantial value, AI has been incorporated in pilot projects but its true applications have yet to see the light of day. In this study, the drivers and barriers to successful AI implementation in the banking sector is analyzed using a panel data of 28 semi-structured interviews with AI experts in the field of banking and finance. AI-oriented role models and process capabilities were revealed to be essential prior to having the trained algorithms reach the level whereby the AI applications can run devoid of human involvement and moral trepidations.

## 1.2 Purpose

Banks are constantly forced to transform their operations in order to stay relevant in a complex and competitive sector. To do so, the key is in maintaining customer loyalty which includes addressing the aspects of customer trust, satisfaction, commitment and perceived value [10]. Constant improvements on customer service and the use of advanced technologies can redefine the processes of banking services as proven by Google and Facebook. Yet, many conventional banking services providers fail to provide the needed flexibility and innovative capabilities. Hence, FinTechs are deemed as the more viable breakthrough to conventional banking service sectors [11]. FinTechs skip on legacy architectures and instead use advanced technologies along with lean and agile procedures to produce improved customer positioning, reduced costs and accelerated innovations speeds. They have catalyzed major innovations in diverse areas including wealth management, payment, lending and crowdfunding [11] as well Solid State Technology Volume: 63 Issue: 5 Publication Year: 2020 6402 Archives Available @ [www.solidstatetechnology.us](http://www.solidstatetechnology.us) as stimulated AI and machine learning adoption in banking services [12]. The highly customer-oriented FinTech organizations have spurred traditional banking services to transform. The incessant prevalence of computed data between customers and banks demonstrate a huge prospect for assessments, analysis and product endorsements. Conversational interfaces are now being reshaped from a rigid and complex form-based communication to a conversation-oriented communication due to the redesigning of digital experiences; in short, AI enables a more natural communication between consumers and banks via writing, conversations and gestures [13]. Additionally, the digital natives i.e. the customer segment from 1995 forwards [14] want the banking service providers to give them similar experiences with that offered by Microsoft, Facebook and other digital suppliers including user-friendly financial products, refined information organization, enhanced procedures with shorter throughput, and product/service customization pre- and postpurchase [14]. Based on the interviews with banks software vendors in this study, an emphasis was clear on the accretion.

## 2.LITERATURE SURVEY

### 2.1Existing problem

#### *Drivers of AI Adoption Technological Drivers for AI Adoption:*

Some of the interviewed experts state AI adoption is inhibited by the non-availability and poor quality of training data. Poor data quality inhibits one inherent characteristic of AI i.e. its algorithm learns from being exposed to input and output data samples, rendering its need for large quantities of training data. Based on insights gathered from the interviews, there is an inadequate amount of digital data to enable the proper training of an AI system, those that are available are protected by data privacy policies. This lack of digital AI training data is also caused by the analogical set-up of many banking service providers; in short, they still rely on paper. Some of the experts' point to poor market overview as the inhibitor to data quality improvement, which is a surprising revelation as AI engines are now being offered by Solid State Technology Volume: 63 Issue: 5 Publication Year: 2020 6404 Archives Available @ [www.solidstatetechnology.us](http://www.solidstatetechnology.us) all major software providers including Amazon, IBM Watson and Microsoft Azure. Many of the providers in turn enjoy the same market visibility as the start-ups.

### 2.2References

1. <https://www.financialdirector.co.uk/2019/10/03/ai-for-financial-directors-and-cfos/>
2. Aazhvaar, V. (2019). ARTIFICIAL INTELLIGENCE IN INDIAN BANKING SECTOR: CHALLENGES AND OPPORTUNITIES. International Journal of Advanced Research, April 7(5), 1581-1587.
3. Alam, M., & Khokhar, R. (2006). Impact of Internet on Customer Loyalty in Swedish Banks. J. Econ. Psychol. Apr 7;16:311-29.
4. Ardito, L., Petruzzelli, A. M., Panniello, U., & Achille, C. (2019). Towards Industry 4.0. Business Process Management Journal , Bradford Vol. 25, Iss. 2, pp: 323-346.
5. Awad, R. (2011). Considerations on Cloud Computing for CPAs. The CPA Journal , New York Vol. 81, Iss. 9, Sep pp: 11-12.
6. Ayachit, M. M. (2017). ICT Innovation in Indian Banking Sector: Trends and Challenges. IOSR Journal of Business and Management (IOSR-JBM), PP 21-27.
6. Bellman, R. (1978). An Introduction to Artificial Intelligence: Can Computers Think? San Francisco: Boyd & Fraser Pub. Co.
7. Bidgoli, H. (2018). Cloud Computing Deployment: What Have We Learned from Real Life Implementations and Practices. Journal of Strategic Innovation and Sustainability , West Palm Beach Vol. 13, Iss. 1, Jul pp 36- 52.

## 2. Statement Definition

### Customer Problem Statement Template:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

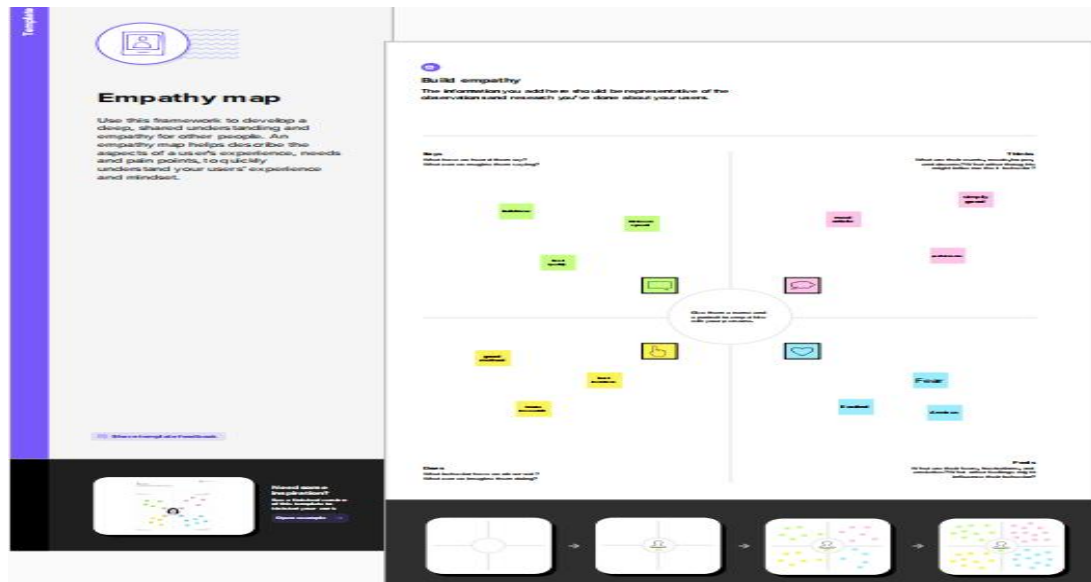
Reference: <https://miro.com/templates/customer-problem-statement/>

### Example:



# 3.IDEATION & PROPOSED SOLUTION

## 3.1Empathy Map Canvas



## 3.2 Ideation & Brainstorming

### Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: <https://www.mural.co/templates/empathy-map-canvas>

Step-1: Team Gathering, Collaboration and Select the Problem Statement

→

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

Template

## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare

🕒 1 hour to collaborate

👤 2-8 people recommended

📄 Share template feedback

Need some inspiration?

View a featured session or browse past sessions to get some ideas.

[Open example](#) →

→

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

Tip

You can select a sticky note and hit the word cloud to instantly create word clouds

spread the love

love	spread	the
spread	the	love
the	love	spread
love	spread	the
the	love	spread
love	spread	the

write

write	write	write
write	write	write
write	write	write
write	write	write
write	write	write
write	write	write

write

write	write	write
write	write	write
write	write	write
write	write	write
write	write	write
write	write	write

Supern

Supern	Supern	Supern
Supern	Supern	Supern
Supern	Supern	Supern
Supern	Supern	Supern
Supern	Supern	Supern
Supern	Supern	Supern

Negatives

Negatives	Negatives	Negatives
Negatives	Negatives	Negatives
Negatives	Negatives	Negatives
Negatives	Negatives	Negatives
Negatives	Negatives	Negatives
Negatives	Negatives	Negatives

Person 6

Person 6	Person 6	Person 6
Person 6	Person 6	Person 6
Person 6	Person 6	Person 6
Person 6	Person 6	Person 6
Person 6	Person 6	Person 6
Person 6	Person 6	Person 6

Person 7

Person 7	Person 7	Person 7
Person 7	Person 7	Person 7
Person 7	Person 7	Person 7
Person 7	Person 7	Person 7
Person 7	Person 7	Person 7
Person 7	Person 7	Person 7

Person 8

Person 8	Person 8	Person 8
Person 8	Person 8	Person 8
Person 8	Person 8	Person 8
Person 8	Person 8	Person 8
Person 8	Person 8	Person 8
Person 8	Person 8	Person 8

Need some inspiration?

View a featured session or browse past sessions to get some ideas.

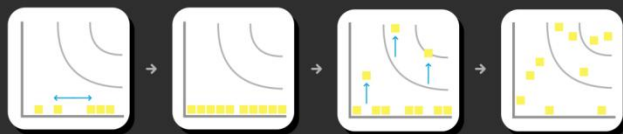
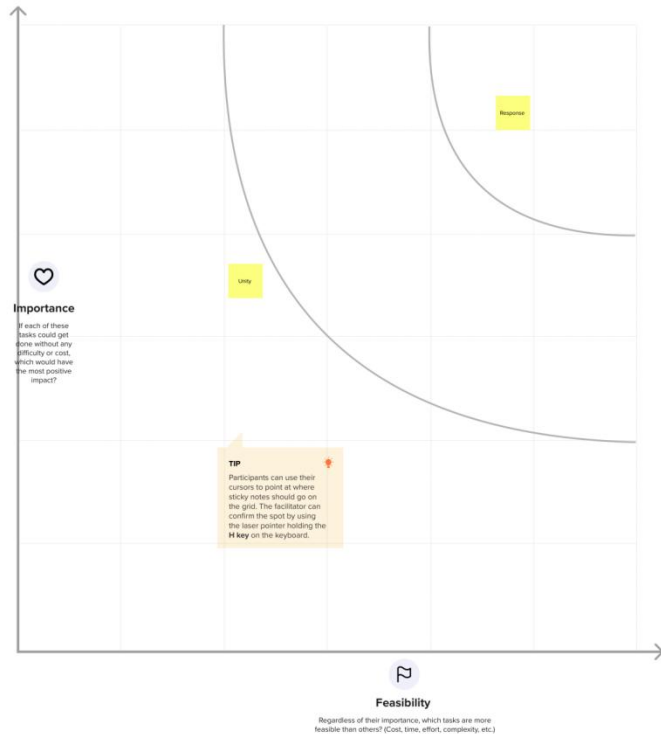
[Open example](#) →

### Step-3: Idea Prioritization

## Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

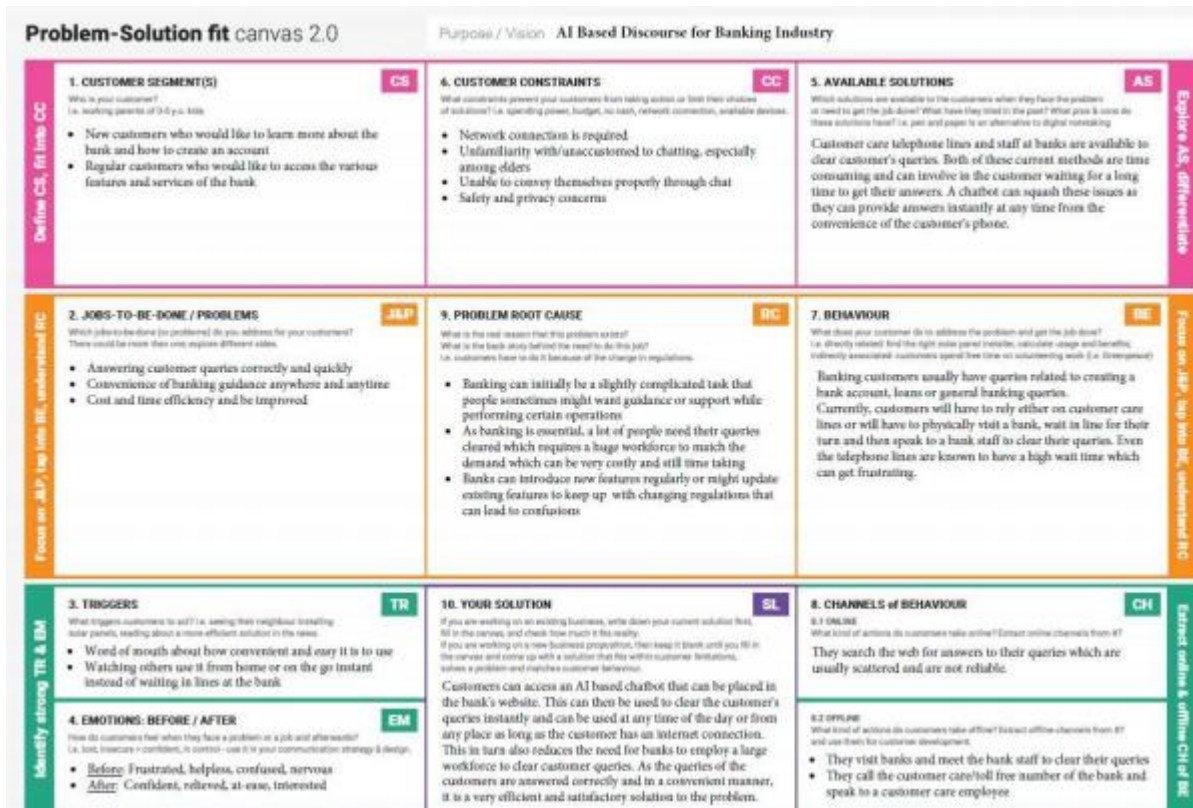


### 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Banks are not able to resolve the queries of customers at all times related to the products or services in satisfactory way which in turn hinders the customer satisfaction. Customers need to visit banks frequently for simple queries.
2.	Idea / Solution description	In order to guide the customers throughout all the financial services provided by the bank, an intelligent system has to be introduced to provide people with the best solution possible.
3.	Novelty / Uniqueness	Chatbots developed using AI should be able to answer any general banking queries on account creation, loan, net banking, other services etc. It addresses the queries of customers immediately and effectively in a cost efficient manner.
4.	Social Impact / Customer Satisfaction	In order to attain the user satisfaction issues associated with banking services, chatbot will provide personal and efficient communication between the user and the bank. It is built to be the overall virtual assistant that can facilitate customers to ask banking- related questions without visiting the bank or calling up customer service centers as well as providing them with relevant suggestions.
5.	Business Model (Revenue Model)	Employing a chatbot will be a cost-effective solution to clear customer queries for banks. It eliminates the need for a massive customer care workforce and even reduces the workload of the bank employees whose efforts can be used elsewhere.
6.	Scalability of the Solution	AI Chatbots provides 24/7 service to clear all customer queries and guide them through all the banking processes. It supports voice assistance feature and maintains a confidential conversation with customers. It can be scaled as per the requirements of the bank to include answers to queries related to any new feature or service introduced by the bank.



## 3.4 Problem Solution fit



## 4.REQUIREMENT ANALYSIS

### 4.1 Functional requirement

Functional requirements define what a product must do, what its features and functions are.

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions. For example:

The system sends an approval request after the user enters personal information.

A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.

**Independent.** This means that you can schedule and implement each user story separately. This is very helpful if you implement **continuous integration** processes.

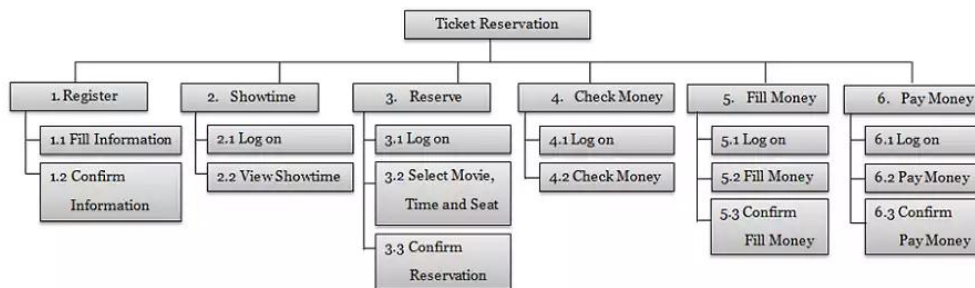
**Negotiable.** This means that all parties agree to prioritize negotiations over specification. This also means that details will be created constantly during development.

**Valuable.** A story must be valuable to the customer. You should ask yourself from the customer's perspective "why" you need to implement a given feature.

**Estimable.** A quality user story can be estimated. This will help a team schedule and prioritize the implementation. The bigger the story is, the harder it is to estimate it.

**Small.** Good user stories tend to be small enough to plan for short production releases. Small stories allow for more specific estimates.

**Testable.** If a story can be tested, it's clear enough and good enough. Tested stories mean that requirements are done and ready for use.



## 4.2 Non-Functional requirements

Nonfunctional requirements describe the general properties of a system. They are also known as quality attributes.

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform. Some examples are:

The website pages should load in 3 seconds with the total number of simultaneous users <5 thousand.

The system should be able to handle 20 million users without performance deterioration. Here's a brief comparison and then we'll proceed to a more in-depth explanation of each group.

Here, we'll just briefly describe the most typical nonfunctional requirements.

### *Usability*

Usability defines how difficult it will be for a user to learn and operate the system. Usability can be assessed from different points of view:

Efficiency of use: the average time it takes to accomplish a user's goals, how many tasks a user can complete without any help, the number of transactions completed without errors, etc.

Intuitiveness: how simple it is to understand the interface, buttons, headings, etc.

Low perceived workload: how many attempts users need to accomplish a particular task.

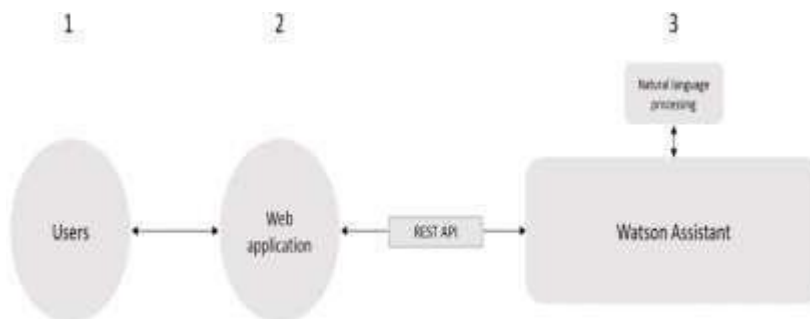
## **5.PROJECT DESIGN**

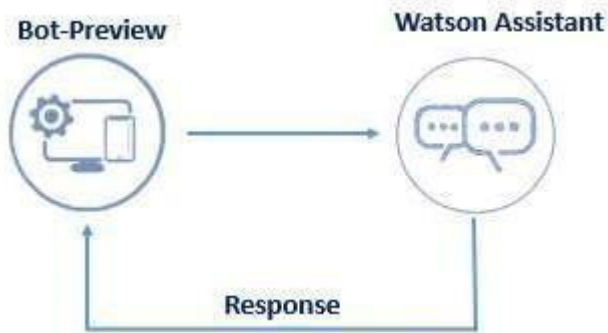
### **5.1 Data Flow Diagrams**

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right

amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.





## 5.2 Solution & Technical Architecture

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	<b>Savings Account Related Actions</b>	<ul style="list-style-type: none"> <li>Type of Savings Account Creation Details</li> <li>Interest Rate</li> <li>Minimum Balance</li> <li>Debit Card</li> <li>Credit Card</li> </ul>
FR-2	<b>Current Account Related Actions</b>	<ul style="list-style-type: none"> <li>Type of Company</li> <li>Current Account Closure Steps</li> <li>Update GSTIN</li> <li>Zero Balance Current Account</li> </ul>
FR-3	<b>Loan Account Related Actions</b>	<ul style="list-style-type: none"> <li>Type of Loan</li> <li>How long for approval</li> <li>Available Loan Amounts</li> <li>Loan Status</li> <li>Joint Loan</li> </ul>
FR-4	<b>General Queries Related Actions</b>	<ul style="list-style-type: none"> <li>Bank Working Days</li> <li>List of Braches</li> <li>Storage Locker Facility</li> <li>Currency Conversion Facility</li> <li>CIBIL</li> <li>Find a nearest branch</li> </ul>
FR-5	<b>Net Banking Related Actions</b>	<ul style="list-style-type: none"> <li>Login Steps</li> <li>Change Net Banking Password</li> <li>Daily Limit</li> <li>Types of Fund Transfer</li> <li>Add Beneficiary</li> </ul>

### 5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (Mobile user)	Download the database	USN-1	As a user, I can register for the application by entering my email, and password, and confirming my password.	I can access my account / dashboard
	Register	USN-2	As a user, I can register for the application by entering my email, and password, and confirming my password.	I can receive a confirmation email & click
	Login	USN-3	As a user, I will receive a confirmation email once I have registered for the application	I can register & access the dashboard with Facebook Login
	Querying	USN-4	User query with a chatbot for clarifications.	
Customer (Web user)	The functional requirements are same as a mobile user	Same as a mobile user	Same as a mobile user	Same as a mobile user

# 6.PROJECT PLANNING & SCHEDULING

## 6.1 Sprint Planning & Estimation

Planning and Estimation are essential in software projects to achieve predictability, reduce the risks involved, and set a basic expectation for all stakeholders. Planning brings a lot of focus on preparation and forecasting whereas Estimation is a process to forecast project-related variables i.e., effort, scope, schedule, etc.

**Planning:** Planning is required irrespective of the project management methodologies that the team follows, whether it is Waterfall or Agile. Planning gives the project team a perspective on how to meet the objective in a systematic way and helps project stakeholders to keep a tab on the project progress and investments done.

As Mike Cohn defines it, “Agile planning balances the effort and investment in planning with the knowledge that we will revise the plan through the course of the project. An agile plan is one that we are not only willing but also eager to change”. This concept exists mainly to avoid the weakness of the planning.

These weaknesses include:

- Focusing on activities instead of delivered features
- Ignoring the prioritization
- Ignoring the existence of uncertainty
- Giving commitments based upon estimations

Such weaknesses in planning make the team not able to cope with the project dynamic environments. For resolving that, the planning must advance to become Agile as well.

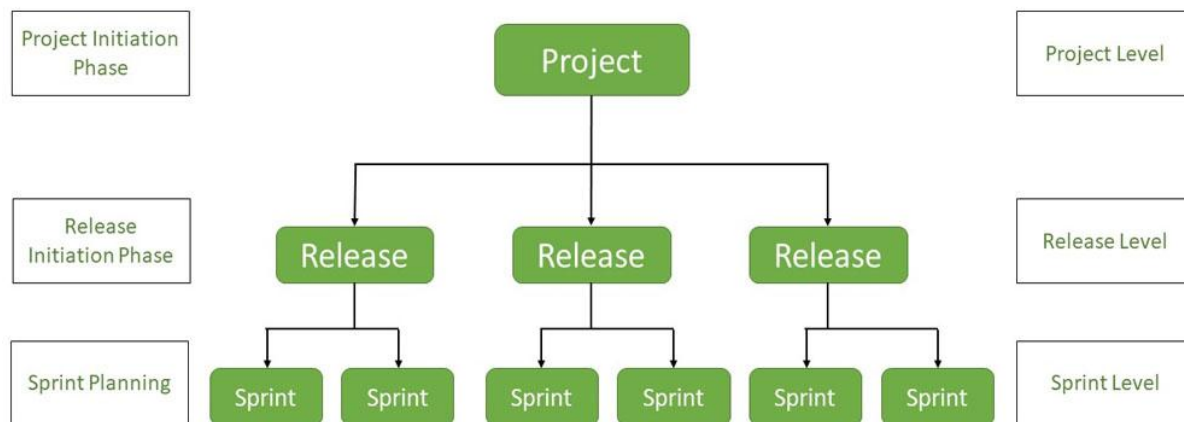
**Estimation:** Schedule, Scope, Cost, and Effort are the four major variables that typically control Software projects. Any changes in any of these variables can have an effect on a project. Estimation is a process to forecast these variables to develop or maintain software based on the information specified by the client.

There are three main challenges faced during estimation i.e., Uncertainty, Self-knowledge, and Consistency of Method used for Estimation. Usage of standardized and scientific estimation methods for estimating size, effort, and schedule, helps towards maintaining minimal variance between the planned estimates and actual values thereby achieving maximum estimation accuracy. This provides a better client experience. All estimation needs for a project cannot be determined by a single method. It is important to have different methods of estimation for different stages.

Planning and Estimation in Agile projects bring a lot of focus on preparation and forecasting. Both these activities are done keeping business context in mind and measurable value delivery is committed to the client. Therefore, it is recommended to have required planning and estimation in Agile from the start of the project, in order to ensure better risk coverage and higher predictability.

Planning and Estimation in Agile projects are generally done at three different stages, which are:

- Project Initiation Level
- Release Level
- Sprint Level (i.e., an iteration within each Release)



Let us now understand each stage in detail.

## Project Level:

### Planning:

Planning activities at the Project level takes 360 degrees of project view in terms of understanding the big picture, overall vision, roadmap, value delivery, planning for better risk coverage, creation of Product Backlog, defining the Releases, etc. This is the highest level of planning that focuses on the impending activities of the project and its ultimate goal. The Product Owner (or Client/Business manager) is the primary stakeholder involved in this activity.

During this stage, the Product Owner (PO) explains to the project/delivery team the strategies which need to be followed by them to deliver the potentially shippable product on time. The team's role here would be to understand the strategic plan clearly and fully fill the needs of the Product Owner to the fullest.

Once the project goal is known, then it is required to define how to achieve that goal. It's all about breaking down the requirements into various segments so that the final deliverables can be built, validated, and released. This helps to plan and estimate various releases as per the feature requirements. At this point, the Product Owner's role becomes crucial. This planning leads to the generation of Product Backlog, which then is used in further planning i.e., Release Planning. Ignoring the Project Planning activity would lead to developing an end product that is not useful or not as expected by the stakeholders.

In practice, planning at the Project level involves a lot of components that provide an overall control in order to achieve predictability in each delivery, mitigating the risks, and setting the basic expectation with all stakeholders. These components are related to infrastructure planning, quality management planning, environment setup planning, tools and reuse planning, build automation and continuous integration planning, etc. However, at a high level, the important aspects of planning include the creation of a Product Backlog, having Requirements

Workshop, creation of a Release plan by defining the releases, and planning of Agile practices through Application Life Cycle Management tools, that need to be adopted in the project.

- **Product Backlog:** Product Backlog is a prioritized list of simplified requirements which is necessary to accomplish the intended delivery of the product/feature. It is a key input for planning. Product Owners often prepare the backlog with the consultation of business strategists and end-users. The prioritization is often based on business value, release date, interdependency and is maintained in a commonplace such as centralized MS Excel file, ALM tools, etc.
- **Workshops:** The main purpose of conducting workshops is for gathering, understanding, and prioritize requirements, and defining a high-level plan for the Releases mainly during the Project Initiation phase. A workshop is an event in which a group of people collaborates to present various ideas in order to achieve a defined goal with the help of an impartial facilitator. It is not the same as brainstorming. In a workshop, there should be a clear idea of what is required and what should be the outcome.
- **Application Life Cycle Management:** Application Lifecycle Management (ALM) aids in managing Agile projects by providing mechanisms to deliver software more predictably and to drive business value. Agile ALM tools are used for planning and estimating user stories and sharing them within the team. It can be used for building a Product Backlog, Sprint Backlog, establishing team commitment and velocity, visualizing team activity and project progress via Burn Down charts, and reporting on team progress. Just by dragging up/down user stories in the backlog, the team can prioritize them in order. Also, teams can efficiently capture, self-assign, and manage their tasks using a good ALM tool.

### **Estimation:**

During the Project Initiation stage, functional requirements are at a high level, or the details of requirements are not well-formulated and documented. In the Product Backlog, many requirements are still at the Epic or Feature level. At this point, estimation is required as it will help in the prioritization and planning of the Releases. Different estimation techniques can be used at this stage. Some of them are

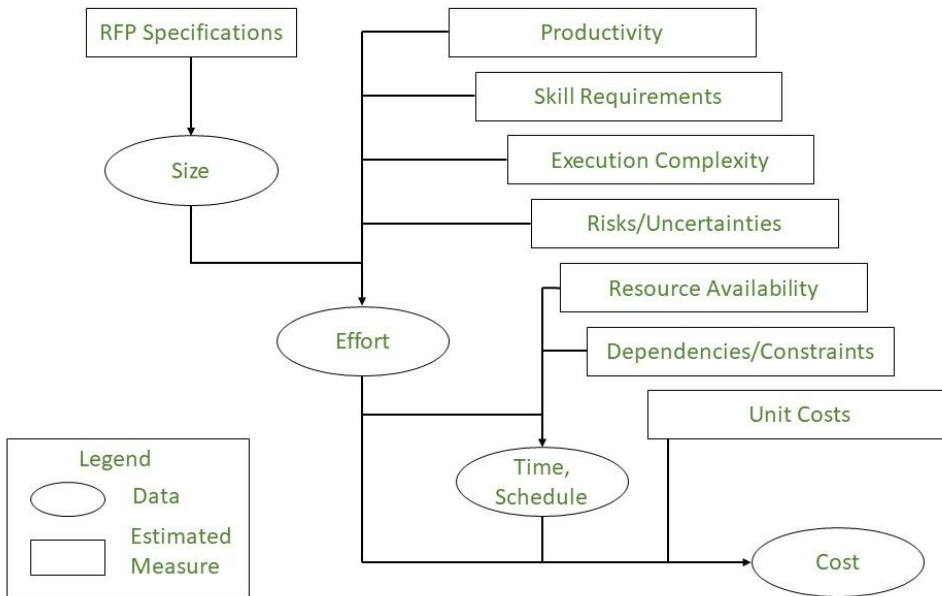
- QFPA (Quick Function Point Analysis)
- Use Case Points
- Complexity based estimation
- COCOMO (Constructive Cost Model)
- COSMIC FP

During the Project Initiation stage, estimation should be done as a group activity, where results should be compared and disagreements are resolved. All the project assumptions and risks should be highlighted clearly. Typically, the team for performing this activity will include:

- Product Owner
- Clients & Business Stakeholders
- Project Manager
- Developers and Designers
- Testers

The general approach for estimation at the project level is illustrated by the diagram below:





The Project Level estimate will give details about the

- The approximate number of Releases/Sprints that may be required before deployment
- Approximate end date of the project
- Approximate staffing plan
- Approximate effort plan etc.

The project-level estimates get refined regularly at the planning phase of each Release/Sprint.

## Release Level:

### Planning:

Without a release plan, the project team would be bouncing from one Sprint to the next, unsure of where to stop (i.e., logically). A shippable product is delivered and ready to be deployed at the end of each Release. As a result, it is critical for a project team to comprehend and adhere to the Release plan.

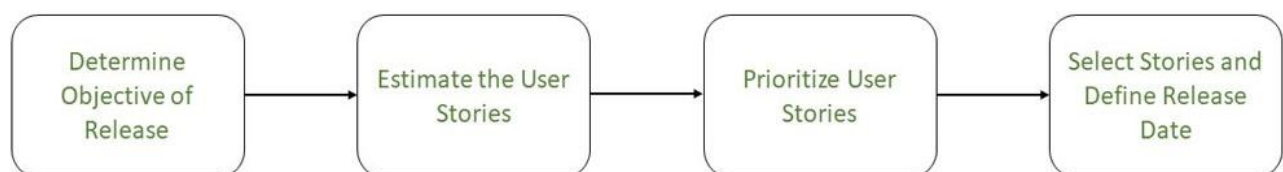
Release planning is all about planning and scheduling user stories (i.e., functional/non-functional requirements) that the team will turn into working software and deliver for that release. The team decides on the number of Sprints (duration 1 to 4 weeks) for each Release in a time-boxed manner where a set of user stories get converted into the working software while undertaking Release Planning.

The release plan can also mention some key assumptions like how long a Sprint will be, how big the team will be, who will be working in the team, the starting of the first Sprint, the end of the last Sprint, etc. Releases are recommended to be planned anywhere between three to six months based on business context for the annuity (long duration i.e., longer than a year) projects, and for the non-annuity projects, one to three months.

It is important to plan for the Releases, as it provides a clear picture to all the stakeholders, on what is expected to get into production, in due course of project execution. Release planning is an important element as it takes place before the start of each new release. This helps the team to deliver the expected project output on an incremental basis to the client which helps them to get early feedback so that future planning gets more aligned to the need of the client.

All stakeholders identify and agree on the duration, goal, and content of each Sprint and Release. However, because requirements may be added or removed as the project advances, at the beginning of each Release and Sprint, the planning for the same is revisited and the scope is revised. The project team can decide to have additional Sprints or Releases during the course of the project depending on the dynamics at that point in time.

Following are various activities involved in Release planning:

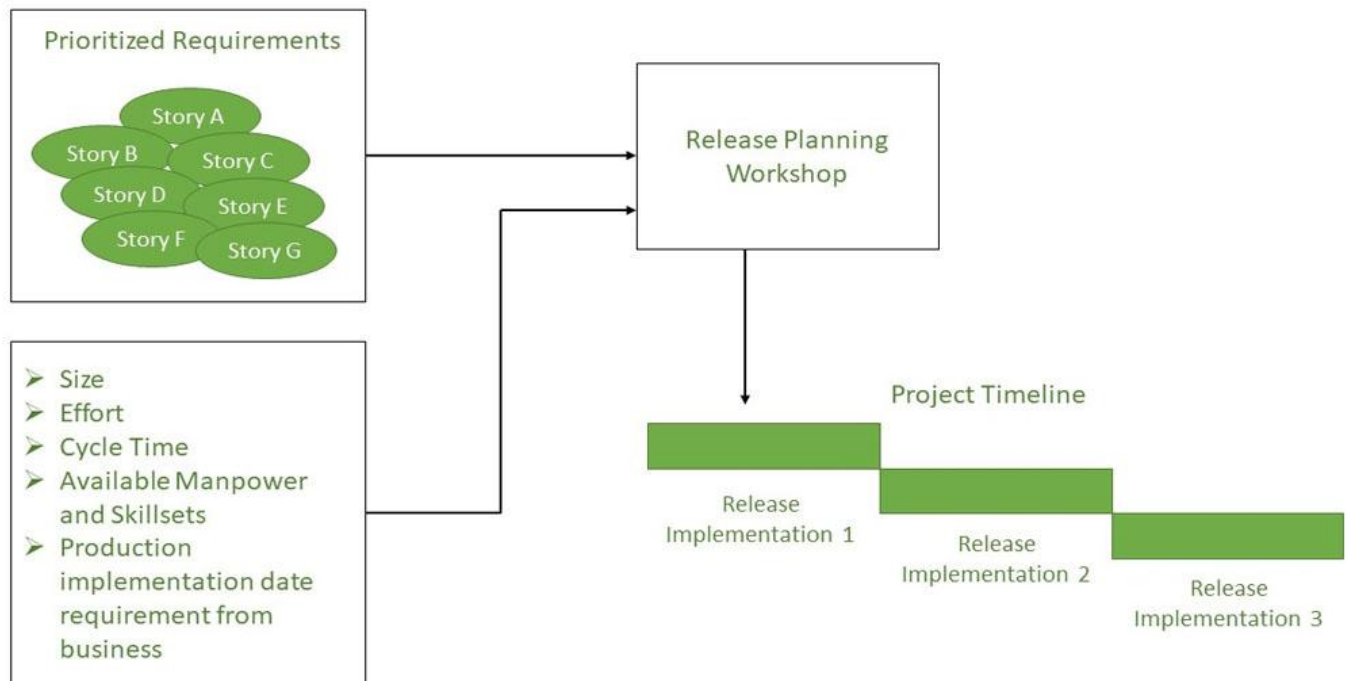


- **Determine Objective of Release:** The Product Owner will generally discuss the desired objective in the Release planning meeting. Mostly the objective will be either time\date driven output or functionality-driven output. When the target is time\date driven output, the Product Owner will expect the release to be finished by said date. Product Owners will usually set a target for completing 'x' functionalities by the end of the Release.
- **Estimating the User Stories (Size):** Product owners will always have a wish list of items that they would like to be targeted during every upcoming Release of the project. It is important to identify and estimate the size (complexity) of the User Stories based on the business value associated with them. These estimates provide key inputs that need to be considered for the Release planning to define the timelines for the Release as well as for the duration of the Sprint.
- **Prioritizing User Stories:** Projects always face a situation where there is either too much work to deliver or too less time to deliver. So, it is always advisable that the product owner prioritizes the user stories available, so that team can work on it accordingly based on the business necessity.
- **Selecting stories and defining Release date:** By this step, the agile team would have information about User Story estimates (in size/complexity), the prioritized user stories (based on business necessity), and the required duration for each Sprint. These inputs would help in Release planning which meets the objective as defined by the Product Owner (or Client) for a Release. All members of the project team, along with the Product Owner and any other stakeholders (i.e., end-users, etc.) are involved in this activity and the Release end date is defined collaboratively with a common consensus.

#### **Estimation:**

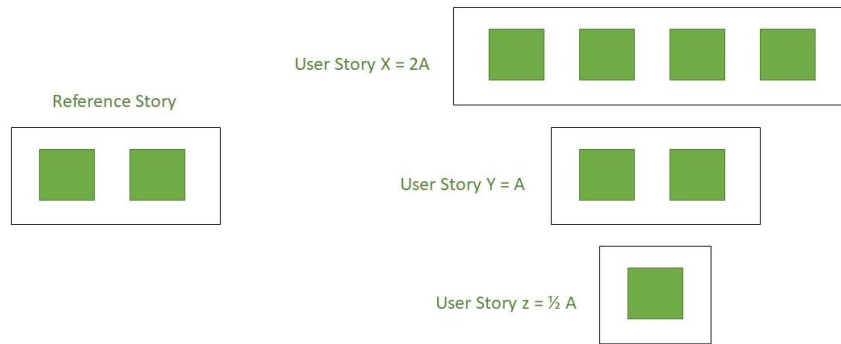
Release level estimation is done at the time of Release planning. The prioritized requirements are taken from Product Backlog which is in the form of User Stories. The User Stories are estimated in terms of Story Points during Release planning which focuses on estimating the

size of the software to be delivered for that particular Release. The other inputs considered are project-level size, effort, cycle time, and available skills. Based on this estimation, a number of releases and a total number of story points in each release are planned for the overall project.



Estimation during Release Planning is done by the entire team using one of the following techniques:

- Wideband Delphi (or Consensus approach) follows the steps while performing the estimation of user story points using the Wideband Delphi method:
    - The coordinator discusses the User Stories with the team and the Product Owner.
    - Each team member estimates the number of person-days needed to complete each functionality.
    - The coordinator compares and analyses the individual estimations with the team.
    - The process is repeated until there is no significant difference in the estimation by the group.
    - This exercise can usually be done in three rounds.
  - Estimation by Analogy (or Extrapolation) follows the steps while performing the estimation of user story points using the Estimation by Analogy method:
    - A user story's estimation is based on its relationship with one or more references.
    - User stories of similar size are collected together and estimated as a group.
    - For example, a user story "Ability to add payee" is estimated to take 40 hours to complete. If the user story "Ability to edit payee" is half as complicated as "Add payee," the effort required for this story will be 20 hours.
- Following is another depiction of an example:



- Planning Poker technique uses estimation cards, which are based on the Fibonacci series i.e. 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, and  $\infty$  while estimating the story points for the User Stories. Planning poker includes all the team members, i.e. developers, testers, analysts, etc. It's a team-wide consensus-based relative sizing technique. Following are the steps followed while performing estimation using this method:
- Each member/estimator has a deck of cards with a legitimate estimate on each card.
- The moderator (who does not usually play) reads a story, which is then briefly discussed.
- Any questions that are raised are answered by the Product Owner.
- Each estimator chooses a card that represents their estimate and puts it facing down (hidden).
- The cards are all turned over at the same time so that everyone can see them.
- If the estimates differ, the highest and lowest estimators might be used to justify the difference.
- Re-estimate until the story points are agreed upon.

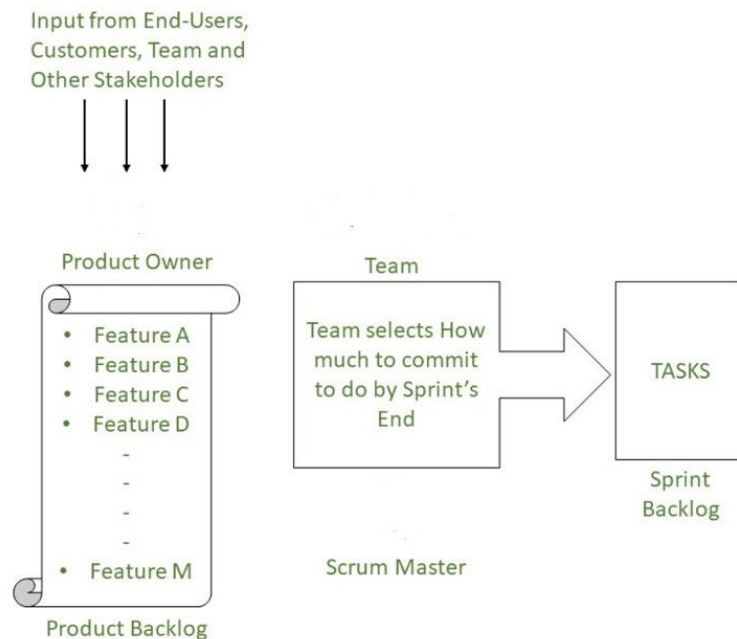
## Sprint Level:

### Planning:

The release plan provides a high-level view of what the team intends to build, and what the team intends to deploy at the end of the Release. It does not provide a detailed view of how the team would plan to drive the work within Sprints. Once Releases and their Sprints are known, the team starts planning for each Sprint. This occurs at the beginning of each Sprint, i.e., on the first day of Sprint, and allows the team to be more explicit about what they will deliver at the end of the Sprint. The team considers the Scrum Master's suggestions as well as the Product Owner's prioritized list. The team then decides what they can take from the prioritized backlog for this Sprint, then breaks down the tasks for each user story and assigns it to themselves.

In the Sprint Planning meeting, the team will be focused more on what they need to do, to complete user stories selected for that Sprint. This meeting is attended by the Product Owner, Scrum Master, all the team members like analysts, programmers, testers, database engineers, user interaction designers, and so on. The output of the Sprint planning can be recorded in a

simple spreadsheet or an ALM tool (Application Lifecycle Management), where the team will have all the user stories of that Sprint and its tasks mentioned sequentially (i.e. Sprint Backlog).



Ideal Time Estimation is based on the Bottom-Up Approach where the business requirements are broken down into low-level activity by the team members and each activity is estimated separately. Team members are asked to sign up for tasks and estimate how long it will take them to complete them in hours or days.

When a team estimate using ideal time, they are referring to the time required for a programmer to complete a feature or task in comparison to other features or tasks. The team is aligned to accomplish the expected goal by displaying the effort remaining in a commonplace (i.e., Burndown Charts).

Individual team members select an activity and submit estimates for ideal time estimation of tasks. If there is a disagreement in these estimates among the team members, then they discuss it and come to a consensus.

## 6.2 Sprint Delivery Schedule

Since sprints take place over a fixed period of time, it's critical to avoid wasting time **during planning and development**. And this is precisely where sprint scheduling enters the equation.

In case you're unfamiliar, a sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.

## Create Sprint Schedule:

The product owner typically determines the duration of the sprint and checks with the team to make sure it aligns with its workloads and resources.

While there may be multiple project heads collaborating on a sprint, it's ultimately important to have one owner who oversees all aspects of sprint planning. Likewise, there should be one single schedule to avoid confusion and keep projects running according to a set plan.

Teams often run into trouble when they create more than one schedule. This can create conflict and derail projects midway through their cycles. To ensure things stay on track, one schedule makes sense.



\*For Next Sprint

## 6.3 Reports from JIRA

JIRA offers reporting in a number of different formats. Project reports that are available from the home screen of the selected project, Gadgets that can be added and arranged in Dashboards and for each filter, the issue navigator offers various output formats that can be used in third party reporting software. Additionally, we will mention some advanced methods that customers have been using.

### Standard Reports

In JIRA, a project will automatically offer standard reports available to the user without any necessary configuration. These standard reports comprise a wide range of reporting applications such as time tracking, workload and also abstract reports like Pie Charts that can be used in various ways.



## 7.CODING & SOLUTIONING (Explain the features added in the project along with code)

### 7.1 Feature 1

#### Purpose of Having Coding Standards:

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

#### 1.Limited use of globals:

These rules tell about which types of data that can be declared global and the data that can't be.

#### 2.Standard headers for different modules:

For better understanding and maintenance of the code, the header of different modules should



follow some standard format and information. The header format must contain below things that is being used in various companies:

- Name of the module
- Date of module creation
- Author of the module
- Modification history
- Synopsis of the module about what the module does
- Different functions supported in the module along with their input output parameters
- Global variables accessed or modified by the module

### **3.Naming conventions for local variables, global variables, constants and functions:**

Some of the naming conventions are given below:

- Meaningful and understandable variables name helps anyone to understand the reason of using it.
- Local variables should be named using camel case lettering starting with small letter (e.g. **localData**) whereas Global variables names should start with a capital letter (e.g. **GlobalData**). Constant names should be formed using capital letters only (e.g. **CONSDATA**).
- It is better to avoid the use of digits in variable names.
- The names of the function should be written in camel case starting with small letters.
- The name of the function must describe the reason of using the function clearly and briefly.

### **4.Indentation:**

Proper indentation is very important to increase the readability of the code. For making the code readable, programmers should use White spaces properly. Some of the spacing conventions are given below:

- There must be a space after giving a comma between two function arguments.
- Each nested block should be properly indented and spaced.
- Proper Indentation should be there at the beginning and at the end of each block in the program.
- All braces should start from a new line and the code following the end of braces also start from a new line.

### **5.Error return values and exception handling conventions:**

All functions that encountering an error condition should either return a 0 or 1 for simplifying the debugging.

## **7.2 Feature 2**

### **Advantages of Coding Guidelines:**

- Coding guidelines increase the efficiency of the software and reduces the development time.



- Coding guidelines help in detecting errors in the early phases, so it helps to reduce the extra cost incurred by the software project.
  - If coding guidelines are maintained properly, then the software code increases readability and understandability thus it reduces the complexity of the code.
  - It reduces the hidden cost for developing the software.
1. **Avoid using a coding style that is too difficult to understand:**  
Code should be easily understandable. The complex code makes maintenance and debugging difficult and expensive.
  2. **Avoid using an identifier for multiple purposes:**  
Each variable should be given a descriptive and meaningful name indicating the reason behind using it. This is not possible if an identifier is used for multiple purposes and thus it can lead to confusion to the reader. Moreover, it leads to more difficulty during future enhancements.
  3. **Code should be well documented:**  
The code should be properly commented for understanding easily. Comments regarding the statements increase the understandability of the code.
  4. **Length of functions should not be very large:**  
Lengthy functions are very difficult to understand. That's why functions should be small enough to carry out small work and lengthy functions should be broken into small ones for completing small tasks.
  5. **Try not to use GOTO statement:**  
GOTO statement makes the program unstructured, thus it reduces the understandability of the program and also debugging becomes difficult.

Here is an example of the utility class.

```
/**
 * Utilities for manipulation with domain objects.
 *
 * @author radek.hecl
 */
public class DomainUtils {

    /**
     * Prevents construction.
     */
    private DomainUtils() {
    }

    /**
     * Null safe copy date function.
     *
     * @param source source date, can be null
     * @return copy of the source date or null if source is null
     */
    public static Date copyDate(Date source) {
        if (source == null) {
            return null;
        }
        return new Date(source.getTime());
    }

    // ... other static methods
}
```

### 7.3 Database Schema (if Applicable)

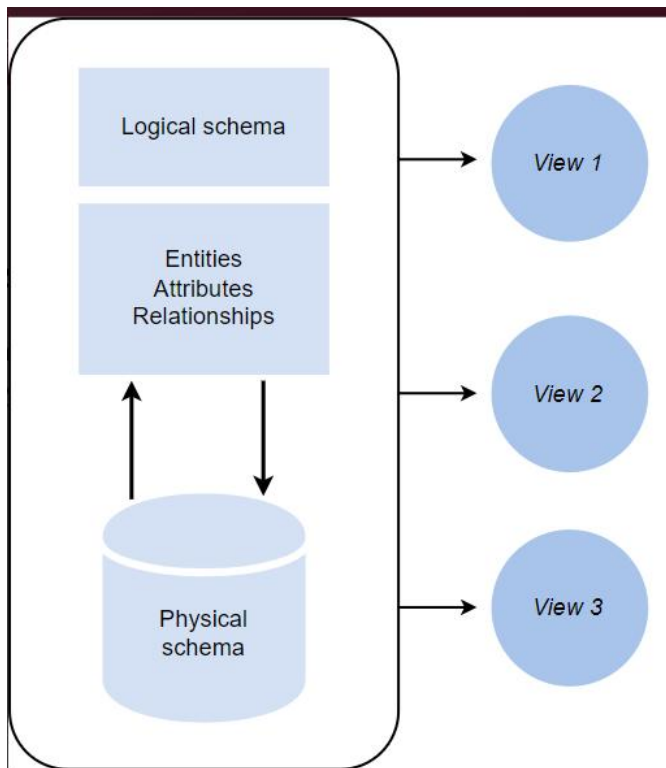
A **database schema** is an abstract design that represents the storage of your data in a database. It describes both the organization of data and the relationships between tables in a given database. Developers plan a database schema in advance so they know what components are necessary and how they will connect to each other.

**A database schema will include:**

- All important or relevant data
- Consistent formatting for all data entries
- Unique keys for all entries and database objects
- Each column in a table has a name and data type

#### Database schema types

There are two main database schema types that define different parts of the schema: **logical** and **physical**.



## 8.TESTING

### 8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into

### 8.2 User Acceptance Testing

#### 1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

#### 2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

#### 3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3

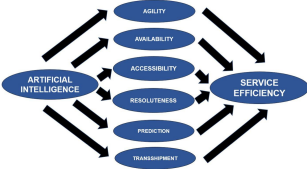
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

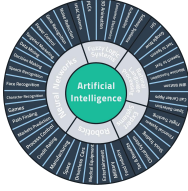
## 9. RESULTS

### 9.1 Performance Metrics

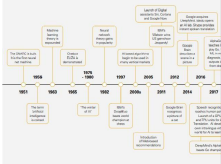
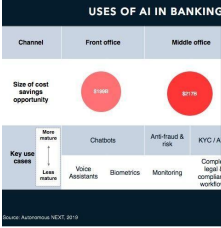
#### Model Performance Testing:

Project team shall fill the following information in model performance testing template.

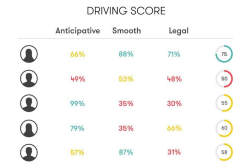
S. No.	Parameter	Screenshot / Values
1.	Dashboard design	No of Visualizations / Graphs -
2.	Data Responsiveness	
3.	Amount Data to Rendered (DB2	No of scene Added

	Metrics)	
4.	Utilization of Data Filters	
5.	Effective User Story	No of Scene Added -
6.	Descriptive Reports	No of Visualizations / Graphs -

Project team shall fill the following information in model performance testing template

S. No.	Parameter	Values	Screenshot
1.	Model Summary	-	
2.	Accuracy	Training  Accuracy -  Validation  Accuracy -	

3.	Confidence	Class	
	Score (Only	Detected -	
	Yolo Projects)	Confidenc	
		e Score -	



## 10.ADVANTAGES & DISADVANTAGES

### Advantages of AI for Banking:

AI can help the bank understand the expenditure pattern of the customer, The bank can come up with a customized investment plan & assist the customers for budgeting, banks can send the notification about the advice for keeping a check on the expenses and investments based on the data, The transactional & other data sources can be tracked to help understand the customer's behavior and preferences to improve their experience.

Artificial intelligent can sift through massive amounts of data and identify patterns that might elude human observers, One area where this capacity is particularly relevant is in fraud prevention, Artificial intelligence and machine learning solutions are deployed by many financial service providers to detect fraud in real time.

and mobile banking become increasingly popular as a tool for 24/7 transaction, AI enables Banks to access customer data, such as detailed demographics, website analytics & records of online and offline transactions, machine learning can integrate & analyze information.

Risk assessment process while giving loans requires both accuracy & confidentiality, It is a very complex & critical process, Artificial intelligence can handle & simplify this process by analyzing relevant data of the prospective borrower, Artificial intelligence can combine & analyze data related to the latest transactions, market trends, and the most recent financial activities to identify the potential risks in giving the loan.

Banks must be bankable for presenting secure & swift transactions, Artificial intelligence is designed to detect the fraud in the transactions on the basis of a pre-defined set of rules, the mobile app can detect any suspicious activity in the customer's account on the basis of behavior analysis, any online transaction of a huge amount from the customer's account which has a history of small transactions can be detected instantly.

Artificial intelligence plays a vital role in protecting personal data, As we witness a rapid rise in the instances of cybercrimes, AI-based fraud detection can prevent such attempts, So, for the banking and finance sector, AI has a tremendous scope in the domain of cybersecurity, The mobile app development services can detect the issue of fraud & data breach for the banks.

## **Disadvantages of AI for Banking:**

The production & maintenance of artificial intelligence requires high costs as they are very complex machines, AI consists of advanced software programs that require regular updates to meet the needs of the changing environment, In the case of critical failures, the procedure to reinstate the system and recover lost codes may require enormous time & cost.

Although Artificial Intelligence can learn & improve, it still can't make judgment calls, Humans can take individual circumstances and judgment calls into account when making decisions, something that AI might never be able to do, Replacing adaptive human behavior with AI may cause irrational behavior within ecosystems of humans & things.

AI can offer a lot of power to the few individuals who are controlling it, so, AI carries the risk and takes control away from humans while dehumanizing actions in several ways, Artificial Intelligence delivered to wrong hands can turn out to be a serious threat to humankind, If individuals start thinking destructively, they can generate havoc with these advanced machines.

Artificial intelligence allows you to replace the workforce with machines that can lead to wide-reaching unemployment, if the use of AI becomes rampant, people will be highly dependent on the machines & lose their creative power, Be it banking or any other sector, AI can increase the unemployment rate, Individuals with nothing to do can lead to the devastating use of their minds.

## **11.CONCLUSION**

banking holds a crucial role in our day-to-day life. We must adhere to the banking system as responsible citizens.

## **12.FUTURE SCOPE**

Artificial intelligence, or AI for short, is encountered in every aspect of our lives and the world we live in. Robots and machines are starting to perform certain tasks even better than us by deep-learning and mimicking human intelligence and actions.

- While 50% of respondents say they know a little about AI, only 10% consider themselves well-trained in AI.
- While 32% of respondents say they are worried about AI, 30% are enthusiastic and 27% are hopeful.
- 40% of respondents say they expect AI to be much smarter than them.

When we turn to the banking industry, it is more likely to talk about the benefits of AI in general. The role of AI in the banking industry is to enable banking services to run much more efficiently, securely and promptly, rather than taking jobs from people.

This is just the beginning, because AI powered credit scoring has the ability to easily analyze much more complex data.

Robotic process automation produces very important outputs to reduce operational costs.

## 13.APPENDIX

### Source Code

```
import pickle
import os
import pathlib
class Account :
    accNo = 0
    name = "
    deposit=0
    type = "

    def createAccount(self):
        self.accNo= int(input("Enter the account no : "))
        self.name = input("Enter the account holder name : ")
        self.type = input("Ente the type of account [C/S] : ")
        self.deposit = int(input("Enter The Initial amount(>=500 for Saving and >=1000 for
current"))
        print("\n\n\nAccount Created")

    def showAccount(self):
        print("Account Number : ",self.accNo)
        print("Account Holder Name : ", self.name)
        print("Type of Account",self.type)
        print("Balance : ",self.deposit)
```



```

def modifyAccount(self):
    print("Account Number : ",self.accNo)
    self.name = input("Modify Account Holder Name :")
    self.type = input("Modify type of Account :")
    self.deposit = int(input("Modify Balance :"))

def depositAmount(self,amount):
    self.deposit += amount

def withdrawAmount(self,amount):
    self.deposit -= amount

def report(self):
    print(self.accNo, " ",self.name , " ",self.type," ", self.deposit)

def getAccountNo(self):
    return self.accNo
def getAcccountHolderName(self):
    return self.name
def getAccountType(self):
    return self.type
def getDeposit(self):
    return self.deposit

def intro():
    print("\t\t\t*****")
    print("\t\t\tBANK MANAGEMENT SYSTEM")
    print("\t\t\t*****")

```

```
input("Press Enter To Continue: ")
```

```
def writeAccount():
```

```
    account = Account()
```

```
    account.createAccount()
```

```
    writeAccountsFile(account)
```

```
def displayAll():
```

```
    file = pathlib.Path("accounts.data")
```

```
    if file.exists ():
```

```
        infile = open('accounts.data','rb')
```

```
        mylist = pickle.load(infile)
```

```
        for item in mylist :
```

```
            print(item.accNo, " ", item.name, " ",item.type, " ",item.deposit )
```

```
        infile.close()
```

```
    else :
```

```
        print("No records to display")
```

```
def displaySp(num):
```

```
    file = pathlib.Path("accounts.data")
```

```
    if file.exists ():
```

```
        infile = open('accounts.data','rb')
```

```
        mylist = pickle.load(infile)
```

```
        infile.close()
```

```
        found = False
```

```
        for item in mylist :
```

```
            if item.accNo == num :
```

```

        print("Your account Balance is =",item.deposit)
        found = True
    else :
        print("No records to Search")
    if not found :
        print("No existing record with this number")

def depositAndWithdraw(num1,num2):
    file = pathlib.Path("accounts.data")
    if file.exists ():
        infile = open('accounts.data','rb')
        mylist = pickle.load(infile)
        infile.close()
        os.remove('accounts.data')
        for item in mylist :
            if item.accNo == num1 :
                if num2 == 1 :
                    amount = int(input("Enter the amount to deposit : "))
                    item.deposit += amount
                    print("Your account is updted")
                elif num2 == 2 :
                    amount = int(input("Enter the amount to withdraw : "))
                    if amount <= item.deposit :
                        item.deposit -=amount
                    else :
                        print("You cannot withdraw larger amount")

    else :
        print("No records to Search")
    outfile = open('newaccounts.data','wb')
    pickle.dump(mylist, outfile)

```

```
outfile.close()
os.rename('newaccounts.data', 'accounts.data')
```

```
def deleteAccount(num):
    file = pathlib.Path("accounts.data")
    if file.exists ():
        infile = open('accounts.data','rb')
        oldlist = pickle.load(infile)
        infile.close()
        newlist = []
        for item in oldlist :
            if item.accNo != num :
                newlist.append(item)
        os.remove('accounts.data')
        outfile = open('newaccounts.data','wb')
        pickle.dump(newlist, outfile)
        outfile.close()
        os.rename('newaccounts.data', 'accounts.data')
```

```
def modifyAccount(num):
    file = pathlib.Path("accounts.data")
    if file.exists ():
        infile = open('accounts.data','rb')
        oldlist = pickle.load(infile)
        infile.close()
        os.remove('accounts.data')
        for item in oldlist :
            if item.accNo == num :
                item.name = input("Enter the account holder name : ")
                item.type = input("Enter the account Type : ")
```

```

        item.deposit = int(input("Enter the Amount : "))

    outfile = open('newaccounts.data','wb')
    pickle.dump(oldlist, outfile)
    outfile.close()
    os.rename('newaccounts.data', 'accounts.data')

def writeAccountsFile(account) :

    file = pathlib.Path("accounts.data")
    if file.exists ():
        infile = open('accounts.data','rb')
        oldlist = pickle.load(infile)
        oldlist.append(account)
        infile.close()
        os.remove('accounts.data')
    else :
        oldlist = [account]
    outfile = open('newaccounts.data','wb')
    pickle.dump(oldlist, outfile)
    outfile.close()
    os.rename('newaccounts.data', 'accounts.data')

# start of the program
ch=""
num=0
intro()

while ch != 8:

```

```
#system("cls");
print("\tMAIN MENU")
print("\t1. NEW ACCOUNT")
print("\t2. DEPOSIT AMOUNT")
print("\t3. WITHDRAW AMOUNT")
print("\t4. BALANCE ENQUIRY")
print("\t5. ALL ACCOUNT HOLDER LIST")
print("\t6. CLOSE AN ACCOUNT")
print("\t7. MODIFY AN ACCOUNT")
print("\t8. EXIT")
print("\tSelect Your Option (1-8) ")
ch = input()
#system("cls");

if ch == '1':
    writeAccount()
elif ch == '2':
    num = int(input("\tEnter The account No. : "))
    depositAndWithdraw(num, 1)
elif ch == '3':
    num = int(input("\tEnter The account No. : "))
    depositAndWithdraw(num, 2)
elif ch == '4':
    num = int(input("\tEnter The account No. : "))
    displaySp(num)
elif ch == '5':
    displayAll();
elif ch == '6':
    num = int(input("\tEnter The account No. : "))
    deleteAccount(num)
elif ch == '7':
```

```
    num = int(input("\nEnter The account No. : "))
    modifyAccount(num)
elif ch == '8':
    print("\nThanks for using bank managemnt system")
    break
else :
    print("Invalid choice")

ch = input("Enter your choice : ")
```

GitHub & Project Demo Link

<https://github.com/IBM-EPBL/IBM-Project-28781-1660116772>