Quick
Detailed
Just Right

System Development Instant Reference

Nicho Wong

DataFlyer

Cover layout designed by Nicho Wong.

Cover image designed by vectorjuice / Freepik and used with permission.

Nicho Wong. System Development Instant Reference.

Abstract

This book is mainly for IT professional direct and immediate reference. It covers different aspects of system development. It is detailed enough to help them solving the problems they encounter every day.

The reference guide is very concise. Anyone could read them quickly and learn the knowledge in just a short time. If you encounter any special problem, you can refer immediately to the corresponding chapter, including what items need to be checked, what the steps are, what areas need more attention, and what should be done.

It is based on the author's over twenty-five years of software and system development experience in the public and private sectors, most of them have been adopted by large corporation or government in Hong Kong and worldwide for many years. Readers may discover that some procedures are more suitable for projects of a specific scale, but the original intention is for reader to check the reference first, and then customize, modify or even expand the procedures according to own business or development needs.

TABLE OF CONTENTS

III. Common Scenario

i. Old System Issues

- Many bugs or problems encountered
- Existing system difficult to use and not user friendly
- System limitation and hence not flexible for specific daily operation
- Previous project completed in hurry with insufficient time and resource allocated
- Only most serious issues are tackled, all the rest use temporary solution or left unsolved
- Previous vendor didn’t have the sufficient knowledge to handle due to tender awarded only to comparatively better candidate
- Previous staff are left with frustration or even fired due to high pressure under tight schedule with no satisfactory solution come out
- Limited knowledge transfer by verbal conversation, no matter technical or business operation
- Poor documentation or not up to date with some missing or inaccurate information

ii. Common Observation

- Large scale project usually cannot be completed as the original planning with many quality issues found and involves extra human and financial resources.
- Project finished either in extended time or with adjusted scope, worst case got terminated
- For project divided into phases, usually due to delay in first phase, following phases started while first phase not complete. Too many tasked accumulated for next phases to solve and got delay too
- Situation not much difference no matter how large the company size is, where the people is graduated from and what gender or race they are
- Only simple project with experienced staff, well-known technology and familiar business operation has a higher success rate
- Tender complexity is under estimated, too many or too complicated task required with insufficient time allowed
- Tender leave too many rooms for putting additional requirement later
- Over workload due to adoption of multiple approaches at the same time, i.e. cloud and on premises environment, waterfall plus agile
- Time is over spent on minor issues and work, not concentrate on the core and major
- Waste time in too many meeting with few practical uses especially for daily one
- Not well aware to manage and control work to meet deadline
- Time spent on mutual criticization but not focus on finding solution together
- Stakeholder with highest authority involve too late and overthrow existing work
- User know what they really want until the system is visualized or they could try

- User requirement changes are too many and too frequent
- Controversary between user and vendor on whether new requirement is within scope
- Stick to existing approach of work and refuse to change or report even found problem
- Technical difficulties encountered that solution takes time to find out and try
- Improper adoption of technology without much evaluation and too late to change
- Unable to recruit suitable staff but only comparatively better
- Exaggerated control work by monitoring side on vendors for own but not project benefit
- Vendor afraid of negative rating will hide all the problems even better solution could easily be found if discussed with client together
- Project Management Office with inappropriate person appointed will not function well.
- No record of requirement meeting details leading to wrong or missing implementation
- No architect to check the overall system design with different sub-system & component
- Missing hard or soft skills required for successful project management: knowledge, human relationship, facilitation, negotiation, leadership, team building, optimistic and perseverant.
- New members added but catch up takes time cannot solve immediate problem

- **ITIL (Information Technology Infrastructure Library)**

  It provides comprehensive set of best practices for IT service management (ITSM) so as to align IT services with business needs and improve service delivery. It covers the following service management processes:

  - **Service Strategy**: define how services will be delivered
  - **Service Design**: design services and processes to support
  - **Service Transition**: manage change, risk, and quality assurance during service deployment
  - **Service Operation**: ensuring services are delivered effectively and efficiently
  - **Continual Service Improvement**: ongoing improvement of services and processes.

  Framework details are mentioned in the in five core publications with guidelines and best practices.

- **TOGAF (The Open Group Architecture Framework)**

  Widely-used enterprise architecture framework that provides a comprehensive approach for designing, planning, implementing, and governing enterprise information architecture that helps organizations align IT strategy with business goals:

  - **Architecture Development Method (ADM)**
    Step-by-step approach to develop enterprise architecture
  - **Enterprise Continuum**
    Repository of architecture assets used to guide the development of new architectures
  - **Architecture Content Framework**
    Provides detailed model for developing architecture deliverables
  - **TOGAF Reference Models**
    Standardized models that can be used as templates for developing specific architectures

Both CMMI and ITIL is less use for organization now but some company or public organization still adopt it.

It is quite expensive for training, certification, hiring certified professionals or external quality assessor.

iv. Other Approaches

- **PRINCE2 (Projects IN Controlled Environments)**
  Structured approach to ensure projects are well-planned, executed, and controlled.
  - **Defined Roles and Responsibilities**: to ensure accountability and efficiency
  - **Continued Business Justification**: ensure project is viable and beneficial
  - **Focus on Products**: deliver tangible and usable product that meet project objectives
  - **Tailor to Suit the Project**: adapt methodology to fit specific needs and context
  - **Manage by Stages**: to monitor progress and control risks
  - **Manage by Exception**: make decisions within defined tolerances with top management involved only when exceptions occur
  - **Learn from Experience**: apply lessons learned from previous projects to improve

- **OGCIO Large Scale Project Implementation**
  - Tailor makes to unique needs of large-scale IT projects for the HKSAR government
  - Provide detailed guidelines for scoping, planning, and risk management, including complexity assessment, stakeholder analysis, and defining project boundaries and deliverables, risk mitigation and outsource management

Observation

- Follow standard and methodology prevent project from commonly known issues if use and refer to properly but no guarantee since new problems are constantly arise every day.
- Statistics show that the effectiveness of adopting methodology such as Agile seems to vary from project to project, and success or failure seems to be related to managers themselves.
- Quite common that known problem still exists if lack of proper management.
- Better to focus on solving problem and achieve objective rather than purely following procedure
- Adaptation, adjustment or combination of different methods or approaches may be necessary for project success. There are endless solution and possibilities all subjected to own innovation.
- Don't be superstitious on any reference unless it really helps and could solve problem on hand.

V. Requirement Collection

- Involve all stakeholder especially top management that make the final decision
- Invite front end staff responsible for daily operation since they know every details
- Meeting invitation should be sent as early as possible
- If meeting not possible for all, consider sampling survey by questionnaire
- Check with user existing problems, bugs, inconvenience operation, new function preferred
- Check complaint or feedback received so far
- On site observation for particular problem
- Use email or phone call to double check minor details
- Different group of user requirement may be contradicted and need to discuss
- Prototyping such as mock screen is easier for communication
- For large user group, let them to consolidate first to save time if tight schedule or extend time
- Seek permission for voice recording in meeting for double check of details only
- Consider to use AI technology to convert voice to text during meeting if allowed
- Write down requirement in details and send to user for double validation and confirmation
- Set deadline for requirement collection and remind user earlier
- Unless critical or affect normal system functioning, no additional requirement should be added that could delay project schedule seriously
- Requirement after deadline may consider back during implementation especially if critical
- Be responsive to user requirement if within scope, time and only minor effort is involved

...

VI. Development Strategy

- After project is awarded or confirmed, system analyst or programmer could start to research, learn and test core technology that could be used
- Basic coding could start if standard framework such as Spring or Spring Boot is used
- Basic function could be developed first, e.g. system code/ parameters, account, logging, audit trail
- Pilot test core code before implementation to avoid side effect on project such as performance
- Reuse features or functions developed in other projects if possible, .e.g. microservice
- Resue if possible existing database structure or some configuration file
- Multiple projects need to develop same function, assign same person(s) to work on it
- Need architect to check overall technical issue but well communicate with each team leader
- ...

VII. Design Consideration

- Think from the angle of user but not developer or designer
- Minimize learning curve for user
- Reduce user brain burden
- Interface should be easy to understand and convenient to use
- Common Operation
  - Consider 2 factor authentication such as email for login
  - Prevent multiple logins at the same time to protect user account
  - Show outstanding task not finish to user in inbox after login
  - Alert user for tasks that near deadline
  - Remind time could be set on user own for task start, 3/4, middle, 1/4 and due date
  - User’s daily, most use and important function should be placed at most notable place
  - Ensure all use case are handled and no missing
  - Consider the most efficient and effective operation for user
  - Keep everything simple and avoid too complicated steps
  - Streamline the work flow between different parties and not miss important one
  - ...

- Technical Concern
  - Responsive to different device, platform and browser use especially resolution
  - Loading, response time and performance
  - Maximum number of transactions
  - Maximum number of user support
  - Concurrency: prevent dead lock and no overwrite for multiple user access on same record
  - Multilingual support
  - ...

VIII. Implementation Consideration

- Architecture design of what component to have, how they work together and communicate
- Block diagram listing all network nodes and the security control inside, e.g. firewall or proxy
- Calculate system sizing with reference to what data are stored and quantity with 5 years forecast
- Determine CPU core, memory and storage required
- Determine using on premise servers, private/ public cloud or hybrid, hybrid increase project complexity and public cloud need to protect corporate data, especially classified/ sensitive data
- Determination of version control, repository management, deployment and rollout tools
- Use most common technology with maximum support, solution and talent available
- Use stable but not latest version to avoid problem arise with no solution on market yet
- Avoid using complicated technologies that are tied to specific products, organization or staff
- Ask for functional implementation code sample before determination of what tool to use
- ...

XIII. Schedule Control

- List out the task to be done in sequence according to submission deadline of deliverables
- Mark down every task with detailed sub task or work breakdown for better monitoring and control
- Check priority and dependency of all task and readjust sequence if necessary for best arrangement to speed up task completion schedule, i.e. critical path
- Millstone should be highlighted since payment would be paid, e.g. SA&D, UAT or Rollout
- List out target start/end date and then actual start/end date with remark in last row with team member name with special attention to critical task with biggest effect to the project
- Fill in by project manager or team leader (top-down), can ask the team member to fill in estimation themselves based on past experience of similar work (bottom-up) and adjust by system analyst
- Allow sufficient time for project task
- Buffer time cover application, feedback, approval, rejection, amendment, resubmission and effective required days such as firewall setting
- Allow 1 to 2 weeks for user comment but prevent endless loop by user that delay project, especially too many details in requirement collection period that should be during implementation
- ...

XIV. Quality Control

- Early start on researching and acquiring related skill sheet
- Think from user angle what could be improved and how
- Make sure all user scenario and navigation path are covered
- Consider efficient and effective method of implementation
- Check existing best practice and implementation for reference
- Seek help from supervisor, peer, technical forum (local or foreign) if necessary
- Utilize testing tool available in market to detect issue earlier
- Peer review especially with similar experience, could be standby resources in future
- Test major technology earlier for performance or special need
- System architecture, functional specification and network diagram documented for checking
- Checklist or procedures are produced for easy follow-up and ensure no missing items
- Tender and evaluation guideline produced to ensure planned features and functions are available
- Record is for new comers to understand the project and follow up
- Documentation provide sufficient details for quick reference is enough
- ...

XVI. Risk Management

- Evaluate from time to time the potential risk that could affect project schedule, cost & quality
- Estimate the chance of occurrence, i.e. high, medium or low probability
- Estimate the severity of impact, i.e. critical, high or low
- Register and report the risk by matrix to inform related parties immediately but not wait
- Discuss with all responsible parties, seek for advice on how to put the project back on right track
  * Management should never work on own assumption but check with staff advice
- Involve earlier top management making final decision or front-end staff well-known of operation
- Prefer continues & incremental change limiting project scope instead of big change
- ...

III. Identify User

Consideration when designing for users:

- Design based on user need and convenience but not for designer or developer’s own favor
- User information provided online may not be real, e.g. age or gender
- User understanding or interpretation may be different due to background or experience
- Allow user customization on preferences such as language, communication channel or pace
- Consider cultural differences among international users
- Aware of taboos and custom of target user group such as religion
- Simplified term and operation plus guidance or assistance is needed for Kids and Elderly
- Increase convenience & reduce mental burden for user, no need to remember command or code
- No need for user to learn much before they could use
- User operation based on previous experience and may ignore repeated details
- Don’t assume user have time to read manual or check everything in detail
- Don’t ask user to do things that system could do for them
- Different kinds of user:
  - Age, i.e. kid, teenagers, adult, retired
  - Gender
  - Language (One or more)
  - ...

IV. Requirement Collection

- Check tender requirement if available
- Check requirement in depth by
  - Major user interview
  - Survey for users at distant and different location
  - Site visit and observation on actual work and usage
  - Focus group interview especially on front end operations staff
  - Special testing to check user preference and response of different design
- Check every potential usage scenario
- Walk through whole operation process or operation from start to end to check if anything missing
- Produce requirement specification and confirm with user

* Beware user may still have missing items forget to mention that need handling

V. Product Analysis

- Determine the features or functions need by evaluating existing product:
  - Existing feature or function to be kept
  - Problems encountered to be solved, e.g. limitation or inconvenience
  - Bugs to fix
  - Functions that need further enhanced to have more or better feature
  - Simplify or combine existing functions, flow or operation
  - Replacement or removal of unnecessary features or functions
  - Introduce new function or feature to take care of new need
- For product developed for the private market
  - Check strength & weakness by product comparison with reference to market trend
  - ...

IX. Information Design

- Present information, convey meaning and experience to user
- Information is presented with a goal, a theme and in a logical flow
- It involves narrative architecture of storytelling, i.e. structuring and organizing data
- Important Consideration
  - Information should be accurate and clear with fact check from different reliable sources
  - Presentation should be clear, concise and visually engaging
  - Progression from simple to complex, major to minor
  - Content presented is the end and navigation are the mean
  - Assistance to find information by visual hint (color or lines), search or guide
  - Memory reinforcement by briefing before and summary afterwards
- Process
  - I. Collection of data
  - II. Grouping of data by type and categories, e.g. time, location or function
  - III. Ranking of data by importance and use frequency
  - IV. Mapping data into different use and grouping
  - V. Determine presentation method and sequence
    - Information delivered to user steps by steps in sequence
    - Information is grouped and let user to select and explore in depth
    - Only relevant information is delivered to user based on collected or anticipated preference

- Information Arts
  - Information could be presented in the form of text, graphics, sound and video in the form of label, menu, tables, navigation planes, icons, maps, infographics or animation
  - Infographics Design Guideline
    - Graph is common use for information presentation
      - Classification
        - ...

  - Balance
    - Maintain harmony by distributing equal weight to elements, including negative space

- Visual Syntax
  - Structure: reinforce by consistent application of module
  - Predictability: respond to similar pattern & way
  - Efficiency: great production economies
  - Clear Focus: limited module reveals spatial logic
  - Flexibility: dealing with different situations
  - Maintainability and extensibility
  - Unity: tie elements work in concert for communication goal
  - Integrity: focus on goal by emergent form
  - Readability: divide content into manageable subsets
  - Control: predicts areas of interest and ease navigation through composition
  - Proximity: associate strongly with nearby elements
  - Similarity: share basic visual characteristics stronger
  - Continuity: prefer continuous, unbroken contours, closure as complete even with break
  - ...

- Navigation Path
  - Arranged in a way for free exploration or in sequential step
    - Linear and sequential step by step operation could guarantee no procedures are missed
  - Allow free exploration mode for user to check back specific information even sequential
  - Organized by different purpose or functions under category or level
  - Avoid too many levels for menu navigation

- Useful feature
  - Short cut for fastest access
  - Search with wild card and filter to locate information easier and faster
  - Site map and index for faster access to particular information
  - Bookmark or history function to go back visited location

- Clear Indication
  - Show current location to prevent from getting lost
  - Show all navigation choice, e.g. Next, Previous, Exit and Return

Graphics Presentation

■ Design matches with the theme	1	2	3	4	5
■ I like the layout design used in the product	1	2	3	4	5
■ The interface used good color contrast	1	2	3	4	5
■ Text display is clear and easy to read	1	2	3	4	5

Interface Element

■ The interface is pleasant and attractive	1	2	3	4	5
■ Interface elements are consistent in look and functionality	1	2	3	4	5
■ Navigation control is appropriate or relevant to the application	1	2	3	4	5
■ Navigation objects are clear, unambiguous and understandable	1	2	3	4	5

Navigation & Interactivity

■ Operation logic is clear and understandable	1	2	3	4	5
■ All functions are clearly identified	1	2	3	4	5
■ Enough navigation hints is provided	1	2	3	4	5
■ Current location is clearly indicated	1	2	3	4	5
■ There is a clear way to return to the main menu	1	2	3	4	5
■ Navigation method between different screens is predictable	1	2	3	4	5
■ I sometimes don't know what to do next with this software	1	2	3	4	5

  - Reliability
    Fault tolerance and fail over to maintain system availability with loose coupling design to minimize dependency and less subjected to external changes
  - Security
    Protection from threats and vulnerabilities to ensure data integrity and confidentiality
  - Maintainability
    Easy to update, change or extend
  - Interoperability
    Standard protocols and formats to facilitate communication and integration
  - Usability
    Easy to understand and shortest path to operate
  - Effectiveness
    Minimum input and maximum output
- **Enterprise Architecture**
  Strategic framework that aligns organization's business processes, information systems, and technology infrastructure with overall goals & objectives

  - **Features**
    - **Performance**: high performance & reliability to support critical business operations
    - **Scalability**: able to handle increased workloads as business grows
    - **Flexibility**: adapt to business requirement change such as new processes & regulations
    - **Security**: measures to protect sensitive data & ensure normal operation
    - **Maintainability**: easy to enhance or fix for issue or problem found

II. Key Player & Influencer

Follow key player and influencer in order to keep up to date of the latest technical development, including social media used by them, it could be Facebook, Instagram, X, LinkedIn or Weibo.

- **Key Player**
  - **Microsoft**
    - Microsoft Learn
      https://learn.microsoft.com/en-us/
    - Microsoft Events Catalog
      https://events.microsoft.com/
    - Microsoft Research
      https://www.microsoft.com/en-us/research/
  - **Amazon**
    - Amazon Web Service
      https://aws.amazon.com/free/
    - Cloud Computing Training & Classes
      https://aws.amazon.com/training/?nc1=h_ls
    - AWS re:Invent
      https://reinvent.awsevents.com/
  - **Google**
    - Android Developer
      https://developer.android.google.cn/
    - Google for Developers
      https://developers.google.com/

■ **Influencer**

No	**Name**	**Contribution**
1	Adrian Holovaty	Django
2	Andrew Ng	Leader in AI education
3	Ben Treynor	Site reliability engineering
4	Bill Gates	Microsoft Founder
5	Bobby Woolf	Enterprise Integration Patterns
6	Brian Behlendorf	Apache HTTP Server
7	Bruce Schneier	Security technologist & author
8	Craig McClanahan	Apache Struts
9	David Heinemeier Hansson	Ruby on Rails
10	David O. Sacks	PayPal COO
11	David Silver	AlphaGo
12	Demis Hassabis	AlphaGo
13	Diane Greene	Google Cloud
14	Doug Cutting	Apache Hadoop

III. Learning Resources

Other than key player and influencer, there are other resources to learn latest technologies:

- **University Open Course**
  - **MIT OpenCourseWare**
    https://ocw.mit.edu/
  - **EDX**
    https://www.edx.org/
  - **Coursera**
    https://www.coursera.org/
- **Training or Events**:
  - **GitHub Universe**
    https://githubuniverse.com/
  - **Tutorials Point**
    https://www.tutorialspoint.com/index.htm
  - **Frontend Masters Live**
    https://frontendmasters.com/
  - **Esri MOOC Training (GIS)**
    https://www.esri.com/training/mooc/
- **Blog, Forum, Social Media & Websites**
  - **BrightTALK**
    https://www.brighttalk.com
  - **TechInsider**
    https://www.facebook.com/techinsider
  - **TechCrunch**
    https://www.facebook.com/techcrunch

- **Web Server**
  - **Nginx**
    - Most popular web server known for its high performance and efficiency
    - Able to handle a large number of concurrent connections
    - Widely used for serving static content and reverse proxy
    - Supports block server configurations to host multiple domains or applications
    - Supports SSL termination, HTTP basic authentication, dynamic IP-based access control, content caching and real-time performance monitoring
    - **OpenResty**: high-performance web platform based on Nginx's non-blocking I/O model to handle high concurrency and integrates various Lua libraries and third-party modules, allowing developers to build scalable and efficient web applications, web services & dynamic gateways
  - **Apache HTTP**
    - Oldest and most reliable web servers
    - High flexibility with auto-indexing, session tracking, URL rewriting, Gzip compression and virtual hosting features
    - Extensive module support, e.g. mod_rewrite for dynamic URL requests

- Language Features
  - Java Lambda Expressions
    - Simplify implementation of functional interfaces
    - Represent anonymous functions without name to enable functional programming by allowing code to be passed as arguments
  - Reflection
    - Inspect and modify its own structure at runtime to analyze or manipulate classes, methods, fields, and constructors.
    - Process of determining which member fields & methods is available on an object
    - Often associated with Introspection to determine which methods of an object are intended to be accessed by other objects, e.g. getters and setters for variables.
    - Dynamically loading classes at runtime.
    - Inspecting or modifying object properties.
    - Creating instances of classes or invoking methods dynamically.
    - Frameworks like Spring and Hibernate use reflection to handle dependency injection and object-relational mapping.
  - Annotations
    - Provide code metadata to convey information to compiler or runtime environment
    - Built-in Annotations: @Override, @Deprecated, @SuppressWarnings

- **C#**
    - Popular for developing Windows applications and games
    - Influenced by Java and share many similarities, such as syntax, object-oriented concepts, multithreading, automatic garbage collection and extensive libraries to handle file, networking, and data structures
    - Support Language Integrated Query (LINQ), which provides powerful query capabilities of data sources directly within the language, e.g. Object, XML and SQL
    - Includes features like properties, events, and delegates
- Visual Basic
    - Used in many legacy systems with long history
    - Simple and easy-to-read syntax
    - Developing Windows-based event-driven applications rapidly by drag-and-drop design tools that respond to user events like key presses and mouse action
    - Excellent integration support in Visual Studio

- **Python**
    - Simple and clean syntax that is easy to read and learn
    - No need for variable explicit declaration and type is determined at runtime
    - Platform-independent running on different operating systems with interpreter installed
    - Automatic and manual memory management
    - Extensive Standard Library for web development, regular expressions, I/O, etc.
    - Rich ecosystem of third-party libraries:
        - **Pandas:** data manipulation and analysis with data structures Series & Data Frame
        - **NumPy:** numerical computing that support large multi-dimensional arrays

every aspect of a plot

- NoCode Development
  - Allows normal non-technical users to build applications without writing any code
  - Create applications rapidly by using templates and drag-and-drop pre-built components
  - Create complex workflow to automate processes and manage tasks
  - Often with built-in integrations for popular services or APIs connecting other tools & systems
  - Facilitate rapid prototyping to validate ideas and make adjustments
  - Product: Microsoft Power Automate, Webflow, Zapier (automation), Airtable (data management), Comet and Makerpad

  * Avoid using tools using different terms for users & developers for same functionality, or redefine common technical terms incorrectly that create confusion & misunderstanding easily
  * Non-technical users are sometimes defined by tools using misleading terms citizen developer

- Vibe Coding
  - Programming with the help of AI
  - Describes the problem to solve on hand in details as a prompt to a large language model (LLM), code would be generated automatically and debugged
  - Perfect for rapid prototyping and even instant deployment with a shareable URL
  - Tools: Cursor, Windsurf or online integrated development environment (IDE) such as Replit

VI. JavaScript Libraries

- **Asynchronous JavaScript (Ajax)**:
    - Able to handle multiple tasks without waiting for one to finish before starting another
    - Once request is finished, use callback function to handle back response
- **jQuery**
    - Fast and small JavaScript library with rich feature
    - Simplifies HTML document traversal and manipulation, event handling, animation & Ajax
    - Made JavaScript easier to use
- **Dojo**
    - Open-source JavaScript toolkit to build cross-platform, JavaScript/Ajax-based applications
    - With rich set of utilities for DOM manipulation, event handling, and more
- **Bootstrap**:
    - Front-end framework for developing responsive and mobile-first websites
    - With HTML, CSS, and JavaScript components for forms, buttons, navigation, etc.

Above is less use since other framework libraries are more efficient, powerful and flexible:

- **Node.js**
    - Developed by Ryan Dahl initially in 2009 based on V8 JavaScript engine
    - Cross-platform open-source runtime environment to run JavaScript on server side
    - Key Features:
        - Single-Threaded event loop to handle concurrent connections with high performance
        - Asynchronous and Event-Driven: non-blocking I/O operations
        - Node Package Manager (NPM) with open-source libraries & packages
        - Suitable to build fast and scalable Web Servers, create RESTful and Microservices API services, Real-Time Applications or even create Command-Line Tools (CLI)
- **Express.js**
    - Minimalist and fast framework for Node.js
    - Commonly used for building APIs and web servers.

VII. Design Principles

- **General Coding**
  - **Code Readability**
    - **Use Meaningful Variable and Function Names**: descriptive and reflect purpose
    - **Consistent Formatting**: consistent indentation, spacing, and line breaks
    - **Comments and Documentation**: explain the purpose of complex code blocks
  - **Simplicity**
    - **Simple & Straightforward**: avoid unnecessary complexity & over-engineering
    - **Modular Design**: break your code into small, manageable functions or modules
  - **Reusable**
    - **Don't Repeat Yourself (DRY)**: avoid code duplication but reusable function & module
    - **Use Libraries and Frameworks**: save time not to do everything from start
  - **Performance Optimization**
    - **Optimize Code**: writing efficient code that performs well
    - **Profiling and Benchmarking**: use tools to identify performance bottlenecks
  - **Reliability**
    - **Always Available**: service always accessible no matter how many users at what time

- **User Centered Design**
  - User need, desire and goal
  - Area that gets most attention and concern
  - Task to accomplish
  - Existing operating & navigation habits
  - Preferred way of interaction
  - Assumptions on how things should work
  - Increase motivation to use
  - Minimize effort to achieve a goal
  - Simplify navigation
  - Remove obstacles
  - Provide hint
  - Provide feedback
  - Help and assistance when need
  - Clear and distinct navigation path

- **Pareto Principle**
  - Identify the most impactful tasks or elements that contribute to the majority of the results in order to create more efficient, user-friendly, and maintainable systems
  - Focus on the top 20% of features that will be used 80% of the time by users
  - Identify & optimize 20% of code or processes that consume 80% of system's resources
  - Simplify & improve user interface elements that 80% of users interact with most frequently

- **Singleton Design Pattern**
  - It restricts instantiation of class to exactly one object to coordinate actions across the system
  - Systems operate more efficiently with only one object, or restrict instantiation to certain number of objects, e.g. system with single-gateway object managing connection to old system
  - Implementation:
    - Single Instance that class create as single instance
    - Global method of same instance: global class method that returns reference to instance
    - Private Constructor: declare class constructor as private to prevent new instance creation
  - Property method get Instance:
    i. Only first call create instance

- **Observer Design Pattern**
  - Observer (View) is object to display data to user
  - Subject (Model) is business abstraction modeled from the concerned domain
  - Observer registers Subject of interest and unregisters if no more interest

- **Factory Design Pattern**
  - Handle object instantiation (new) & initialization (constructors) that creates explicit association between creator & created classes
  - Use factory methods to create objects without specify exact class of object to be created

- Common Tools Implementing EIP
  - **Apache Camel**: open-source integration framework that implements EIP
  - **Spring Integration**: framework provides support for EIP within Spring ecosystem
  - **Mule ESB**: enterprise service bus that supports EIP
  - **Red Hat Fuse**: integration platform that implements EIP

- **Service Oriented Architecture**
  - Architectural approach that designs software as a collection of loosely coupled services
  - **Key Concepts**
    - **Service: s**elf-contained unit of functionality that performs specific task or business process that is loosely coupled, reusable, independent deployed & managed
    - **Service Provider**: creates, maintains, publishes service
    - **Service Consumer**: discovers, invokes & uses service based on requirement

  - **Key Components**
    - **Web Services:** implementation based on standards **Simple Object Access Protocol (SOAP)** and **Representational State Transfer (REST)**
    - **Service Bus:** intermediary that facilitates communication between services, often implemented as an **Enterprise Service Bus (ESB)**
    - **Business Process Execution Language (BPEL):** for orchestrating &

- **Functional Programming**
    - treats computation as evaluation of mathematical functions
    - avoids changing state and mutable data
    - use of pure functions, immutability, and declarative programming to make code more predictable, easier to understand, and easier to test
    - Key Concepts
        - **Pure Functions:** always produces same output for same input & has no side effects
        - **Data Immutability:** not change once created and use new data structures for modification to prevent unexpected side effects
        - **First-Class Functions:** they can be passed as arguments, returned from other functions, or assigned to variables to enable higher-order function for more flexible & reusable code
        - **Higher-Order Functions:** take other functions as arguments or return functions as results to facilitate code reuse & composition and easier to build complex functionality
        - **Declarative Programming:** describe what to do rather than how to do using expressions to make code more concise & closer to domain, to improve readability & maintainability
        - **Recursion:** function calls itself to solve smaller instances of same problem
        - Common Functional Programming Languages
            - **Haskell:** pure functional programming language with strong static typing & lazy evaluation.
            - **Lisp:** oldest programming languages that supports functional programming
            - **Erlang:** for concurrent & distributed systems that emphasize immutability & message-passing.
            - **Scala:** combines functional and object-oriented programming features and runs on the Java Virtual Machine (JVM).
            - **F#:** functional-first language running on .NET platform and supports both functional and object-oriented programming.
            - **JavaScript (with libraries like Underscore.js and Lodash):** supports functional programming concepts.

- **Lambda Architecture**
  - Handle large-scale data processing by combination of batch & real-time data processing
  - Key Concepts
    - **Batch Layer**: processes large amounts of historical data and produces batch views using distributed storage systems like HDFS & processing frameworks like Hadoop
    - **Speed Layer**: T deals with real-time data processing to provide low-latency updates using stream processing frameworks like Apache Storm or Spark Streaming
    - **Serving Layer**: merges result from batch & speed layers to provide a unified view of the data and serves processed data to users for querying
  - Generate raw data (logs) continuously or in batches
  - Message queue collects raw data and feeds it to both layers
  - Data is stored durably as an immutable master dataset often in distributed file system
  - Computes accurate, historical views over all data
  - Raw data lives in a fault-tolerant store
  - Processed results go to serving layer
  - Periodically reprocesses entire dataset hourly or daily to update batch views
  - Processes recent data in real time for low-latency insights using in-memory stores such as Redis or lightweight databases for quick access
  - Updates views incrementally as new data arrives to compensate for batch layer delays
  - Merges batch and speed layer outputs for queries using NoSQL databases, indexing systems (Elasticsearch), or analytics platforms (Druid)
  - Provides a unified view, i.e. batch for historical accuracy & speed for recent updates
  - More complex & need to maintain 2 separate codebases for batch & stream processing

X. Framework

Framework provide pre-built libraries, tools, and templates that speed up development and reduce the code need to write.

I. Java

- **Spring**
  - First developed by Rod Johnson in late 2002 for enterprise development, latest version is 6.0.
  - Comprehensive infrastructure support: Object Lifecycle Management, transaction management, data access, messaging, remote procedure call, XML and testing, etc.
  - Main Features
    - **Inversion of Control (IoC)** :
      - Not create object, call services or components directly but define inside configuration
      - **Configure container by XML or Java annotations** such as @Resource, @Inject or **Spring specific annotations**, e.g. @Autowire, @Qualifier and @Primary
      - **Framework control** over implementations of dependencies used by objects
      - **Dependency lookup**: framework look for object with **reference by name or type**
      - **Dependency injection**:
        -object/function receives other objects/functions required instead of internal creation
        -**container passes objects via constructors, properties or factory methods**
      - **Autowiring**: **auto resolve & inject** collaborating beans (dependencies) into class without requiring explicit configuration
      - Disadvantage: code difficult to trace since it separates behavior from construction
    - **Aspect-Oriented Programming**
      - Concept developed mainly by Gregor Kiczales at Xerox PARC
      - Implement **cross-cutting concerns** or functionalities span across multiple modules : log
      - Add behavior to existing code (**advice**) by specifying which code is modified **via pointcut** specification without code modification e.g. add logging function
      - Proxy pattern-based that **configured at run time**
      - **Interception** allows for **public method-execution** on existing objects **at join point**
      - Disadvantage: control flow is obscured and unclear

- **Apache Kafka**
  - Distributed event streaming platform capable of handling trillions of events per day
  - For high-performance log aggregation, real-time analytics, event sourcing, and integrating different systems with real-time data pipelines
  - Distributed architecture with fault-tolerant running as a cluster of servers that can span over multiple data centers
  - Allow horizontal scalability by adding broker nodes to cluster for massive amounts of data
  - Optimized for high throughput that can process large volumes of data with low latency
  - Data persisted on disk & replicated across multiple nodes to ensure durability & reliability
  - Allows multiple producers to publish messages to topics and multiple consumers to subscribe to those topics
  - Streams API for processing data streams in real-time or complex event-driven applications
  - Kafka Connect for connecting external systems, e.g. databases, key-value stores, search indexes, and file systems.

- **Apache Karaf**
  - Built on OSGi framework, which allows for modular and dynamic application development
  - Main Features: shell console, remote access, hot deployment, and dynamic configuration
  - Web management console allows plugin management, debugging, and OSGi monitoring

- **Apache Camel**
  - Integration framework and message-oriented middleware that provides components and patterns for routing and mediation rules
  - Use declarative Java domain-specific language to configure routing and mediation rules

- Alibaba Cloud
  - Dominates in Asia, scalable services tailored to e-commerce, big data & Chinese market
  - Major Services
    - Compute: Elastic Compute Service (ECS), Function Compute
    - Storage: Object Storage Service (OSS), Block Storage
    - Database: ApsaraDB (RDS, NoSQL options like MongoDB)
    - AI/ML: Machine Learning Platform (PAI)
    - Big Data: MaxCompute (data processing)
    - Other: CDN, e-commerce, and IoT solutions

- Private & Hybrid Cloud
  - Private clouds are built sometimes due to superior security requirement by keeping data within a dedicated environment to reduce the risk of unauthorized access and data breaches
    - Use cloud management platform to manage cloud infrastructure, e.g. OpenStack
    - Implement automation tools to manage virtual machines, resources and configuration, e.g. Ansible, Puppet, or Chef
    - Use container orchestration tool that automates deployment, scaling, and management of containerized applications, e.g. Kubernetes (K8s)
  - Hybrid clouds use public cloud for non-sensitive tasks but keep critical data on private cloud

- Cloud Native
  - Design & deployment of applications for cloud environments
  - Applications take full advantage of cloud 's scalability, elasticity, resilience, and flexibility
  - Key Aspects
    - **Microservices Architecture**
      - Application is divided into smaller, loosely coupled & independent services
      - Services can be deployed and scaled out individually across cloud
      - Flexibility in choice of technology and programming language for each service

        - **Rate Limiting**
            - Prevent abuse and ensure fair usage of the API
        - **Caching**
            - Improve performance and reduce server load
    - **Service Mesh**: infrastructure layer that manages service-to-service communication e.g. **Istio** works with Kubernetes to provide load balancing, service discovery, authentication, and monitoring capabilities
    - **Open Service Gateway Initiative (OSGi):** framework for modular development of Java often used in large enterprise applications that allow applications to be composed of multiple reusable components (bundles) to be deployed in containers
- **Containers**
    - Package code & dependencies by Docker to ensure consistency across environments
- **Orchestration**
    - Manage containers, automate deployment & scaling by tools like **Kubernetes**:
        - **Pods**: smallest & simplest unit in Kubernetes object model to represent run process instance containing containers that share storage or network resources
        - **Services**: abstraction that defines logical set of Pods & access policy that enable communication between different parts of application inside or outside cluster
        - **Deployments**: abstraction manages deployment & scaling of Pods that provide declarative updates for Pods & ReplicaSets to ensure desired state of application
        - **ReplicaSets**: ensure specified no of pod replicas are running at any given time.

- **OLTP (Online Transaction Processing)**
  - Designed for high transaction volume, real-time processing & ensuring data integrity
  - High Transaction Volume supports large numbers of short, online transactions
  - Real-Time Processing to ensure transactions are processed quickly
  - Data Integrity by using mechanisms ACID (Atomicity, Consistency, Isolation, Durability)
- **Object Relationship Mapping**
  - Mapping between objects in application and tables in a relational database
  - Avoid code that are repeated in multiple places with little or no variation
  - Improve productivity when working with database operations
- **Object Oriented Database**
  - Stores data in objects like structured data in object-oriented programming languages
  - Support class hierarchies & inheritance to represent complex relationships between objects
  - Encapsulation of data and behavior
  - Reusable across multiple applications, scalable and suitable for large data

- **Object Storage**
  - Designed to store large amounts of unstructured data, such as media files, backups, and logs with high scalability and durability. It's commonly used in cloud storage solutions.
  - data storage architecture that manages & manipulates data storage as objects
  - includes **Metadata** of information like file type, size, creation date & permissions
  - unique identifier to retrieve object from storage pool
  - often stored with redundancy, ensuring data is protected against hardware

XIII. Internationalization

- **Encoding Standard** defines how characters, symbols, and control codes are represented in binary form for computer processing and storage
- **Unicode**
  - Universal character encoding standard that encompasses worldwide characters & symbols
  - Allows multiple languages to be processed in a single file seamlessly
  - Facilitates global communication and data exchange between different systems & devices
  - Accommodates the addition of new characters and scripts
  - **Code Point:** numerical value that maps to specific character, e.g. code point for 'A' is U+0041
  - **Encoding format**:
    - **UTF-8**: most widely used Unicode encoding format uses 1 to 4 bytes per character:
      - **1 Byte**: ASCII characters in U+0000 to U+007F
      - **2 Bytes**: Characters in U+0080 ~ U+07FF for Greek and Cyrillic characters
      - **3 Bytes**: Characters in U+0800 ~ U+FFFF for Chinese, Japanese, Korean characters
      - **4 Bytes**: Characters in U+010000 ~ U+10FFFF for emojis and ancient scripts
    - **UTF-16**: uses two or four bytes per character
    - **UTF-32**: uses a fixed length of four bytes per character

- **Web3**
  - Blockchain Foundation

    Distributed ledgers (e.g., Ethereum, Solana) store data and execute code across thousands of nodes to ensures transparency, immutability, and no single point of failure
  - Smart Contracts

    Self-executing programs on blockchains automate trustless interactions
  - Decentralized Applications (DApps)
    - Apps with frontends (like Web 2.0) but backends on blockchains
    - Connect via wallets like MetaMask, using crypto for access or payments.
  - Identity and Ownership
    - To replace username with cryptographic key wallet to prove ownership of assets/data
    - Unique tokens NFTs to certify ownership on-chain such as digital art or game items
  - Decentralized Storage

    Store files across nodes to complement blockchain's limited capacity
- **Web 4.0**
  - Not officially define but concept and speculation
  - Ultra-intelligent and Symbiotic Web almost seamless with real life
  - AI-driven web with enhanced capabilities for data analysis, prediction & automation
  - AI assistants by smart devices called Think IoT

- **Graphics and Animation API**
  - **DirectX:** API bypass Windows Graphical Device Interface (GDI) & direct communicate with hardware:
    Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound. Latest version is 12.
  - **OpenGL**: cross-platform graphics APIs for rendering high-quality graphics
  - **Vulkan**: low-overhead, cross-platform API for high-performance graphics
  - **Metal**: Apple's graphics and compute API designed for iOS, macOS, and tvOS
  - **WebGPU**: web graphics API that provide modern GPU access for web applications
  - **WebGL:** JavaScript API for 3D & 2D graphics rendering on web browser
- **Sound and Music**
  - **FMOD**: middleware used for implementing sound effects and music in games with straightforward, DAW-like (Digital Audio Workstation) interface
  - **Wwise**: middleware for dynamic mixing, spatial audio & interactive music
  - **Audacity**: free and open-source audio editing tool
- **Networking**
  - **Photon**: networking solution for creating real-time multiplayer online games
  - **Netcode for GameObjects/ Entities, Mirror**: Unity library for creating client-server network model, cooperative-style or competitive action games
  - **UNet:** Unity's simpler networking solution that lacked scalability & performance, deprecated
- **Artificial Intelligence**
  - **Pathfinding Algorithms**: A* (A-star) like algorithm widely used for finding shortest path between two points especially for Non-Player Character navigation that often involves Waypoints or specific points use to navigate in environment
  - **Behavior Trees**: ideal for complex AI behaviors, decision-making, and hierarchical tasks that using hierarchical structure where nodes represent tasks or actions, can be composite (containing other nodes), decorators (modifying behavior), or leaf nodes (actions or conditions).
  - **Finite State Machines**: suitable for simple and predictable behaviors consist of states that represent specific behavior or action, and transitions that occur based on conditions or events

* Synchronization between video, animation, and sound is indeed crucial for creating high-quality human animations that lip movements need to match with the dialogue.

- Enterprise Mobile Management (EMMI)
    - Enable businesses to provide employees with the flexibility to work from anywhere
    - Set of technologies, processes, and policies used to manage and secure mobile devices
    - Ensuring corporate data remains secure and no allowed applications to be installed
    - Identity and Access Management (IAM) handles authentication, authorization, and single sign-on (SSO) capabilities
    - Mobile Device Management (MDM) handle device enrollment, configuration, security enforcement, and remote data wiping.
    - Mobile Application Management (MAM) manages deployment, updating, and removal of mobile applications on devices. Control app permissions and only approved apps are used
    - Mobile Content Management (MCM) ensure secure access to corporate content like document sharing, editing, and synchronization while ensuring data security
    - Enables organizations to implement Bring Your Own Device (BYOD) policies without the need for company-owned devices
    - Specific EMM services to be installed on-premises or cloud
    - Require paid licenses or subscription
- Weixin/ WeChat
    - Develop lightweight apps called Mini Programs using JavaScript, WXML, and WXSS
    - Development Tools: Use the WeChat Developer Tools for coding, debugging, and testing
    - Access Weixin's APIs for features like payments, social sharing, and user authentication
    - Resources: https://developers.weixin.qq.com/doc/
- Alipay
    - Alipay supports Mini Programs for app development
    - Use the Alipay Developer Platform for tools, APIs, and SDKs
    - Integrate payment solutions, marketing tools, and user management features
    - Resources: https://developers.alipay.com/
- HarmonyOS Next (HongMeng Xingheban)
    - Use integrated development environment DevEco Studio to create apps
    - Supports ArkTS (TypeScript-based language) & ArkUI to build apps
    - HarmonyOS SDK includes tools for app frameworks, system development, media, AI, etc
    - Utilize DevEco Testing for automated testing and debugging
    - Publish your app on Huawei's AppGallery
    - Resources: https://developer.huawei.com/consumer/cn/develop/

XVI. Artificial Intelligence

- **Artificial Intelligence (AI)**
  Computer systems capable of performing tasks typically require human intelligence, such as learning, reasoning, problem-solving, perception, language understanding & decision-making
- **Artificial General Intelligence (AGI)**
  The ability to understand, learn, and perform any intellectual task that a human can with the aim to replicate broad cognitive abilities of humans across a wide range of domains
- **Generative AI**
  AI systems designed to create text, images, music, or video content.
- **Robotic Process Automation (RPA)**
  Automates repetitive tasks and processes to increase efficiency and reduce operational costs
- **Embodied AI**

- **Development**
  - **Turing Test** by Alan Turing in 1950: measure of machine's ability to exhibit intelligent behavior indistinguishable from that of a human
  - **Symbolic reasoning**: Logic Theorist and General Problem Solver
  - **Rule-based system**: Expert Systems that use extensive knowledge bases to simulate decision-making process of human experts
  - **Neural Network**: perceptron that learn from data
  - **Machine Learning (ML)**
    - Data-driven approaches using statistical methods
    - Algorithms designed to learn from data and improve over time
  - **Deep Learning**
    - deep neural networks with many layers
    - process vast amounts of data (Big Data) and perform complex tasks such as image and speech recognition, natural language processing

- Problem: when calculate slope, i.e. gradient of loss function with respect to a parameter, it becomes very small, almost close to zero (vanishing gradient). Hence, during backpropagation (adjusting network), these parameters barely get updated, especially in earlier layers of network where changes are so small as if not learning much at all

- **Convolutional Neural Networks (CNN)**
  - Specialized neural networks designed for processing grid-like data
  - Use convolutional layers to auto learn spatial hierarchies of features adaptively
  - Use sliding filters to scan and detect patterns sch as edges & shapes across the input

- **Generative Adversarial Networks (GAN)**
  - **Overview**: GANs consist of two neural networks, a generator and a discriminator, that work against each other to generate realistic data.
  - Generator creates fake data, while the discriminator evaluates its authenticity, leading to the generation of high-quality synthetic data.
  - Good for creative AI generating images, audio, or text but tricky to train and unstable

- **Long Short-Term Memory (LSTM)**
  - Type of RNN capture long-range dependencies & maintain information for long periods.
  - Use gates to control the flow of information, addressing the vanishing

- **Feedforward Neural Networks (FNNs)**
  - Simplest neural net that data flows straight from input to output with no loops or memory

- **AI Agent**

  Autonomous entity interacts with environment to achieve specific goal or perform task that can perceive surroundings, make decisions based on observations & take actions to influence

  - **OpenManus**
    - Open-source alternative to Manus AI agent, developed by MetaGPT community
    - Available free on GitHub to anyone with basic technical skills.
    - Modular framework that allows users to build customizable AI agents capable of autonomously handling complex tasks, from coding to data analysis

- **Transformer**
  - Neural network model that revolutionized natural language processing (NLP), speech recognition and computer vision with parallel processing and scalability
  - **Key Components**
    - **Self-Attention Mechanism**

      Weigh importance of different words in sentence relative to each other and capture long-range dependencies so as to understand data contextual relationships
    - **Multi-Head Attention**

      Understand complex relationships by capture various aspects of data in parallel and focus on different parts of input data simultaneously using multiple attention heads

        - **AutoGen**
            - Known for its flexibility & ease of use in developing AI agents for various applications.
        - **Semantic Kernel**
            - Focuses on natural language understanding and generation
        - **CrewAI**
            - Emphasizes collaboration between multiple AI agents to solve complex problems
        - **RASA**
            - Popular for building conversational AI agents that interact with users in natural manner
    - **Generative Models**
        - **Variational Autoencoders (VAEs)** generate blurry image by mapping input data into lower-dimensional latent space (encoder), produce probability distribution for latent variables & then reconstructs original data from it (decoder)
        - **Diffusion Models** generate high-quality images by gradual transform noise into meaningful outputs through Forward Diffusion that add noise to input data in steps until becomes pure noise; and Reverse Diffusion using neural network to remove noise & recover original data
    - **Model Context Protocol (MCP)**
        - Open standard by Anthropic to streamline how LLMs connect to external data sources/ tools for building agents & workflows
        - Defines how contextual information is exchanged between client & server, e.g. message framing, linking requests with responses & high-level communication patterns

- **Agent2Agent Protocol (A2A)**
  Google Open protocol that facilitates communication & coordination between multiple AI agents
- **Large Model Systems Organization (LMSYS)**

- **Al Benchmarks**
  - Essential tools to evaluate performance, accuracy & efficiency of AI models
  - Provide standardized tests & metrics to compare different models
  - Common AI benchmarks:
    - **AIME 2025** American Invitational Mathematics Examination test with problems
    - **Chatbot Arena (LMArena):** test crowd-sourced, blind A/B test where users vote on model responses
    - **Geekbench AI:** cross-platform benchmark using real-world machine learning tasks to evaluate on-device AI capabilities & workload performance of CPU, GPU & NPU
    - **General AI Assistants (GAIA):** evaluate performance on real-world tasks with focuses on reasoning, multi-modality handling, web browsing & tool-use proficiency
    - **GLUE/SuperGLUE:** early standard for language understanding, test sentiment analysis, question answering
    - **GPQA (Science): t**est graduate-level PhD questions on physics, chemistry & biology
    - **ImageNet:** vision benchmark on image classification with 1,000 categories & 1.2M images
    - **LiveCodeBench:** test real-world coding tasks from 2024-10 to 2025-02, testing functional code generation

XVII. Internet Of Things

- **Internet of Things (IoT)**
  Network of physical devices, vehicles, appliances, and other objects embedded with sensors, software, and network connectivity, allowing them to collect and exchange data
- **AIoT**: combines Artificial Intelligence (AI) with the Internet of Things (IoT)
- **Usage**: cost-saving automation, gather real-time data for decision making, enhanced customer experience, connect & remote-control applicants, and access information from anywhere
- **Application**
  - Agriculture: guide irrigation
  - Healthcare: patient monitoring and alert for emergencies
  - Industrial Automation: monitor machinery, predict maintenance or optimize supply chains
  - Retail: track inventory and enhance customer experiences
  - Smart Cities: managing traffic patterns and autonomous driving

- **Requirement**
  - Sensors or other devices that collect/transmit data, track location, detect or avoid obstacle
    e.g. temperature, humidity, motion, light, pressure, proximity & gas sensors, camera & GPS
  - Data Processing & Analysis software and hardware that analyze data locally or in cloud
  - User Interface including mobile apps, web dashboards, wearable devices,

- **Financial Technology (Fintech)**
  - Innovative use of technology to deliver financial services and products
  - Application of various technologies like blockchain, artificial intelligence, big data, and mobile applications in providing traditional financial services:
    1. **Digital Payments**: peer-to-peer mobile payment & digital wallets
    2. **Lending**: peer-to-peer lending services and platform
    3. **Investments**: Robo-advisors, online algorithmic trading & cryptocurrency exchanges
    4. **Personal Finance**: Budgeting apps & finance management tools.

- **Blockchain**
  - Decentralized, secure digital ledger technology that records transactions across a network of computers to ensure no single entity to controls it and data cannot be easily altered
  - First introduced with cryptocurrencies Bitcoin by Satoshi Nakamoto
  - Decentralized Finance (DeFi): new financial system enables direct peer-to-peer transactions without traditional bank and offer decentralized lending, borrowing, & trading services
  - Digital Identity Verification: provides secure and tamper-proof digital identities, reducing fraud and enhancing trust in financial transactions
  - Smart Contracts: self-executing contracts with terms directly written into code, transactions execute automatically if met predefined conditions to reduce the need for intermediaries
  - Tokenized Assets: tokenization of real-world assets, e.g. Non-Fungible Token (NFT) to represent ownership or proof of authenticity of unique item or piece of content

- Decentralized Application (DApp)
  - Application runs on decentralized network rather than centralized server, e.g., blockchain
  - Leverages smart contracts to automate processes without intermediaries i.e. self-executing code stored on blockchain

- **Big Data**
  - Huge & complex datasets traditional tools unable to handle with following characteristics:
    - **Volume:** sheer amount of data being generated in tera or petabytes
    - **Velocity:** data is generated & processed at real-time or near-real-time data streams
    - **Variety:** different data types such as text, images, videos or sensor collected data in structured, semi-structured or unstructured format
  - Implementation
    - Identify use cases & data sources
    - Setup & deploy distributed computing framework to handle scale, e.g., Hadoop, Spark
    - Collect and stream data using data ingestion tools such as Kafka for real-time data
    - Store data in data lakes (centralized repository) or NoSQL DB such as Apache Cassandra or MongoDB
    - Analyze data using batch or stream processing techniques
    - Build dashboards or reports for insights by data visualization tool Tableau, Power BI

- **Quantum Computing**
  - Cutting-edge computing that uses principles of quantum mechanics to process information
  - Use quantum bits or qubits that exist in multiple states simultaneously (superposition) and be entangled with each other (entanglement)
  - Unlike classical bits that represent either 0 or 1, qubits can represent both 0 and 1 simultaneously and hence could perform massive parallel computation
  - State of one qubit is directly related to state of another, even if they are separated by large distances and this allows faster and more efficient information processing
  - Use quantum gates to manipulate state of qubits to perform calculations
  - Quantum computers enhance machine learning and artificial intelligence to solve complex problems, factor large numbers, optimize logistics, or simulate molecular interactions at an atomic level and could even break current encryption methods
  - It is built with superconducting circuits, trapped ions, or photons, cooled to

XX. Security

- Check **OWASP Top 10** most critical web application security risks from time to time:
  - **Broken Access Control**: users access data or perform actions they shouldn't be able to
  - **Cryptographic Failures**: incorrect implementation leads to sensitive data exposure
  - **Injection**: attackers inject malicious code into a program
  - **Insecure Design**: design flaws that can be exploited
  - **Security Misconfiguration**: incorrect configuration & settings leave applications vulnerable
  - **Vulnerable and Outdated Components**: use components with known vulnerabilities
  - **Identification & Authentication Failures**: user authentication & session management issues
  - **Software & Data Integrity Failures**: unable to ensure integrity of software & data

- Run **code scanning tools** to identify common issues such as Codacy or SonarQube
- Run free **OWASP Zed Attack Proxy (ZAP)** to scan web applications for security vulnerabilities

- **Apply server setting** to tackle common security issues
  - Hide server and operation system version to prevent exposing server information
  - Disable directory listing to prevent visitors from seeing contents
  - Obtain valid certificate from trusted Certificate Authority (CA) to configure SSL/TLS to use HTTPS communication between server and client
  - Limit Request Size to prevent denial-of-service (DoS) attack

- **XSS Cross-Site Scripting Attack**

  Attackers inject malicious scripts into web pages viewed by other users
  - Issues
    - Stealing user information
    - Tampering with web page content
    - Spreading malware
  - Types
    - Reflective XSS: script reflected off a web server, e.g. in a search result or error message
    - Stored XSS: stored on server, e.g. database, sent to users when request relevant content
    - DOM-based XSS: script executed after modify DOM environment in victim's browser
  - Causes
    - Insufficient input validation
    - Improper output encoding
  - Solution
    - Strict input validation & filtering dangerous characters & scripts
    - Appropriate Output encoding for output data, e.g. HTML entity &specific contexts
      - HTTP security header settings by using Content Security Policy (CSP)
      - Set X-XSS-Protection response header in web server
    - Secure development practices
      - Use secure programming frameworks and libraries to auto handle
      - Conduct code audits and security tests regularly

  * Usually, will escape ore replace special character, e.g.< or > to prevent inserting code

  * To ensure normal display of Asian text, bypass them by checking character code point

- **OAuth**

  Open standard for access delegation to grant applications limited access to user info

  - ◆ Approve interaction on behalf of resource owner or third-party application obtain access on its own behalf
  - ◆ OAuth 2.0 often uses JSON Web Tokens (JWT) as access tokens to securely transmit information between parties in string containing
    - ➢ Header metadata about the token signed using algorithm HMAC SHA256
    - ➢ Payload with actual data called claims in base64-encoded JSON
    - ➢ Signature cryptographic hash created using secret key or public/private key pair to prove the token wasn't tampered with
  - ◆ User logs in, server issues JWT, client sends JWT in HTTP headers with each request, server verifies signature and trusts the payload without hitting a database

- **OpenID Connect (OIDC)**
  - ◆ Simple identity layer on top of the OAuth 2.0 protocol
  - ◆ Extends OAuth 2.0 that uses tokens for access by adding an ID Token
  - ◆ When user login, Relying Party (RP) redirects to the OpenID Provider (OP)
  - ◆ OP sends back ID Token in JWT with your identity info, or optional Access Token for accessing APIs (OAuth-style) and Refresh Token to get new tokens later
  - ◆ RP checks ID Token's signature using the OP's public key to trust it.

- Scenario
  - Basic
    - Check features listed in Tender Requirement
    - Check Requirement Specification on all user stories
    - Make sure all functions listed in functional specification is tested
    - Checking meeting record
    - Check email from user on additional or changed requirement
  - Testing Techniques
    - Boundary: null, minimum and maximum value
    - Type: try different data type, i.e. incorrect value
    - User Group: try different user group with different authorization
    - Concurrency: multiple users performing same function at same time or period

- Data
  - Use automation tool to generate different set of data (often need correction)
  - Apply production down time at non-office hour to test real data (encrypted if necessary)
- Devicess
  - Different configuration, e.g. CPU, memory and storage
  - Different software version installed
  - Different network segment with different firewall or proxy setting
  - Different network environment Wireless and LAN

- Personnel
  - Peer Review especially helpful if outside project

- Tools:
  - Use Junit to check bug earlier

XXII. Debugging

- Difficulties
  - No specific environment or product to check, e.g. payment gateway in non-production platform or product of different version
  - No exact data to test due to security and privacy protection
  - No access to specific platform due to highly sensitive environment, e.g. police department
  - No related equipment such as old machine of low performance configuration

- **Strategy**
  - Check and interpret error messages correctly but sometime error is due to other causes
  - Record down error code returned if any and search on web may help, many times unable to solve problem but provide hints and direction to look & deep dick into the problem further
  - Check and review logical flow steps by steps to find out problem or what is missing
  - Trace potential area first, then narrow down the scope steps by steps until root is traced

- **Logging**
  - Place trace code in function start & exit, record down start & end time to check time spent
  - Record down who, when and what action to take inside the code
  - Record down value changes of specific variable before and after the function
  - Output statement, variables value or error message to console or log file using datetime format
  - Apply log rotate to zip the log file every 3 months at least to save server

- **Techniques**
  - Check both client console, server and event log for error
  - Check whether data value has been written down into database
  - Common issues: typos, scope, libraries, permission, connection, setting or corrupted data
  - Trace if nothing returns after specific external call, e.g. asynchronous call
  - Check if code fall into endless loop
  - Check if any deadlock or value overwrite for concurrent process
  - Check CPU, Memory and storage size usage during function execution
  - **Source Code**
    - Make sure error is caught or report by implement try...catch blocks or similar mechanism
    - Write unit tests using testing frameworks to catch bugs early

XXIII. Performance Tuning

- **Round Trip Time (RTT)**: key performance metric for web application responsiveness that performance tuning target to reduce, i.e. entire process of sending request from the client to the server, and sending back a response from server to client after processing.
- **Analysis**
  - Trace and locate where the problem occurs by following ways:
    - Debug logging to record function start & end time to calculate processing time
    - Use performance monitoring tools to gather statistics, e.g. JMeter, Neoload, etc

- **Server**
  - Scale Up (Vertical)
    - CPU: add or upgrade
    - Memory: add or replace with faster memory

  - Scale Out (Horizontal)
    - Network configuration: load balancing or clustering to distribute load among servers
    - Add server in on premise
    - Server replication under cloud environment
  - Proxy server to provide caching or request only after change or expire
  - Reverse Proxy to distributes incoming client requests across multiple backend servers and compresses responses from backend servers before sending to clients

        - Views are virtual tables that simplify complex queries but performance gains or not depend on underly query efficiency & how DB engine handles it, e.g. indexed views
    - Clustering to distribute load among different server, e.g. Oracle Real Application Clusters
    - Use flashback to quickly recover from human errors by undoing changes at various levels (row, transaction, table, or entire database) without backup restore, e.g. Oracle's Flashback

- **Programming**
    - Close unused object resources to clear memory, i,e, manual garbage collection
    - Use variable binding instead of writing single SQL to reduce script compilation
    - Display loading message and show remaining time to wait
    - Apply asynchronization to avoid keep waiting for other tasks to complete
    - Allows program to perform multiple tasks at the same time by multiple processors or cores
        - Parallel processing to divide task into sub-tasks that can be processed simultaneously
        - Run multiple threads in pool concurrently & use worker thread to offload time-consuming tasks from main thread
    - Caching: page, object, application

XXIV. DevOps

- Philosophy and set of practices that encourage better collaboration between application development (Dev) and system operations (Ops) teams
- Adoption of iterative development, process automation, programmable deployment and maintenance to ensure continuous software delivery with quality
- Common Practice
  - **Configuration Management**
    - Systematic change management throughout system lifecycle to ensure system is configured correctly, tracked, working as expected, update in controlled manner that won't break the system
    - Key Aspects
      - Identify configuration items (CIs), establish baselines of security & performance
      - Control planning, implement change control process, evaluation & approval
      - Status Accounting to track & report status of CIs & change requests with automation
      - Audit to verify that CIs conform to requirements and ensure system integrity
    - Key Components
      - Artifact Repository
        Centralized location to store build artifacts, libraries & dependencies
      - Source Code Repository
        Stores and manages codebases and their history
      - Configuration Management Database (CMDB)
        Store information and track state of hardware, software, networks, and other assets
    - Tools
      - **Ansible**: open-source automation tool uses YAML-based configuration, no agent to install, orchestrate complex workflows across multiple systems, automates repetitive tasks and workflows and secure handling of credentials and sensitive data
      - **Puppet:** open-source CM tool that uses declarative language, enforced by Puppet agent, modular code with detailed reports, monitoring and tool integration capabilities
      - **Chef:** automation platform that transforms Infrastructure as Code (IaC) that manage and automate infrastructure in scalable and efficient manner

XXV. Networking

- Performance
  - Ensure sufficient bandwidth to handle the expected data traffic and avoid bottlenecks by calculating the size of data to be transmitted, numbers of concurrent users and the overhead for data transmission, i.e. request/response lines and headers.
  - Minimize latency to ensure timely data transmission by upgrading to higher-speed connections such as fiber optics, use dedicated link, implement Content Delivery Networks (CDNs) that distribute content closer to users
  - Implement load balancing to distribute traffic evenly across servers and prevent overloads
  - Monitor network performance to identify and address issues promptly
- Reliability
  - **Use** backup servers or multiple network paths to ensure continuous operation if any failures
- Security
  - Use Hypertext Transfer Protocol Secure (HTTPS) to secure data communication between client and server
  - Original protocol Secure Sockets Layer (SSL) for data communications encryption, even 3.0 is now considered insecure, and has been replaced by Transport Layer Security (TLS)
  - TLS 1.3 is recommended for new implementations due to its enhanced security and performance but TLS 1.2 is also considered secure and is widely used

- Storage Area Network (SAN)
  - Enhance storage devices' accessibility to servers, as if locally attached
  - Add or expand storage using disk arrays & tape libraries to accommodate storage needs
  - Offer high-speed data transfer rates by using Fibre Channel (FC)
  - Use Host Bus Adapters (HBAs) to connect servers to SAN
  - Use SAN Switches to manage data flow between using
  - Data mirroring and replication to enhance data protection & disaster recovery capabilities
  - Widely used in data centers and support virtualized environments
  - Centralized management of storage resources
  - Utilize Redundant Array of Independent Disks (RAID) configurations to combine multiple physical hard drives into a single logical unit for data storage
    - Provides fault tolerance by storing redundant copies of data across multiple drives
    - Allowing parallel access and improving overall performance
    - Common RAID Levels
      - **RAID 0 (Striping)**: distributes data evenly across all drives without redundancy
      - **RAID 1 (Mirroring)**: duplicates data on 2 or more drives, fault tolerance & fast read
      - **RAID 5 (Striping with Parity)**: distributes data & parity info across 3 or more drive
      - **RAID 6 (Striping with Double Parity)**: like RAID 5 with additional parity info allowing for fault tolerance against two simultaneous drive failures
      - **RAID 10 (RAID 1 + 0)**: combines mirroring & striping to provides high performance & fault tolerance but requires many drives

XXVI.Server Administration

- **Monitoring and Performance Tuning**
  - System Monitoring**:** check system performance using tools like Nagios, Zabbix, or Prometheus to detect potential issues early. For Windows, check events, system and application error reported.
  - Performance Tuning**:** optimize server performance by adjusting system settings, managing resources, and addressing bottlenecks.
- **Security Management**
  - **User Management:** Managing user accounts, permissions, and access controls to ensure only authorized users can access the system.
  - **Security Patches:** Regularly applying security patches and updates to protect the server from vulnerabilities and threats.

  - **Certificate Installation**: Generate a Certificate Signing Request (CSR) to

- **Software Installation and Updates**
  - **Installing Software:** Installing and configuring server software and applications as needed.
  - **Updating Software:** Keeping server software and applications up to date to ensure compatibility and security.
- **Resource Management**
  - **Disk Management:** Monitoring and managing disk space usage to prevent storage issues.
  - **Memory Management:** Allocating and optimizing memory usage to ensure efficient server performance.

XXVII. Documentation

- For a traditional formal project, following document are usually required:
  - Project Plan
    - Goal and Objective
    - Cost Benefit Analysis
    - Team & Staff
    - Schedule
  - Requirement Catalogue
    - User Role, Post, Rank, Group, List with name, department, address, email & phone
    - Use Case: Operation, Process, Manual or Automatic Procedures, Rules & Regulations

    - Functional Requirement
      - Function Name, ID & Description
      - Type: General, Case, Report & Administration
      - Mode: Online or Batch

    - Non-functional Requirement
      - Page loading time
      - Response time of user action
      - Number of concurrent users: average/ peak

- System documentation serves 3 purposes:
  - Design Verification by client
  - Implementation Guideline for developers or supporting staff to follow
  - System Understanding by takeover or follow-up team

- Reminders
  - Never treat it as homework but need to work seriously
  - Desirable to simplify to contain only necessary and essential information
  - Use free form easy understanding diagram to reduce the text needed

- Requirement
  - Cover all in details & confirm with user to avoid missing or inaccurate implementation
  - Involve all stakeholder and front-end operation staff during discussion
  - Seek for highest authority opinion earlier to avoid overthrow design at final stage
  - Seek for approval before voice recording for any requirement meeting
  - Simple point form list in email cc to all parties could meet the purpose
  - Collect reply of group representative but not circulate one by one to save time

  - Ensure following items are covered during requirement collection:
    - User type, role, group, authority, total number, on-leave resource arrangement
    - Registration, authentication & authorization
    - Daily Operation Business Flow and Approval Process
    - How many concurrent users' maximum at a time or time period
    - Application Form and what data or information to store
    - Interface

        - Inquiry and Search criteria
        - Availability, loading and response time required for each operation
        - What reports to produce at what time including what information

- System Sizing
    - Determining software and hardware requirements over specific period, e.g.5 years
    - Check expected number of users, transactions per second, data type/size/volume
    - Check average & peak loads, loading and response time
    - Calculate memory, storage and network bandwidth, CPU speed and cores required
    - Estimation result will produce a final score and then could check vendor what system model meeting benchmark are available

Chatper 4. Data Migration and Conversion

I. Definition

I. Data Migration

Data from one or multiple system to be migrated to the new system when the data structure of the source and target are the same.

II. Data Conversion

Data from one or multiple system to be recreated in the new system due to different data structure.

II. Source Data

- Database Record

  I. Table and Fields

  All tables and fields other than system tables, those no longer use or without data

  II. Data Type

  i. Text
  ii. Numerical data
  iii. Datetime
  iv. Binary object, e.g. image

- Script

  Database SQL script in existing database that may be reused or need for reference

  e.g. views, triggers, functions or stored procedures

VII. Data Analysis

- Assumption:
  - i. Data are in use
  - ii. Accurate without human error
  - iii. No missing records

  * Need to clarify, and remove those not use and inaccurate data, and add back those missing
- Data Grouping
  - **Static Data**

    Data never change
  - **Dynamic Data**
    - **Base Data**
      - Essential for system initialization, startup and running

        e.g. basic setting, menu options, system parameters or system code
      - Subjected to changed irregularly after function enhancement

- **Data Dictionary Creation**
  - Data structure of the new system should be ready by the development team
  - List out all tables and fields in source and target database, best in Entity-Relationship Diagram

VIII. Relationship Mapping

- Source and target database schema mapping
- Determine data transformation rules:
  - What source tables/fields mapped to which target tables/fields
  - Field data types in old database changed to what data types in the new database
  - Some field need to define default values for new tables during conversion
  - Separate value in source table field to multiple fields on target or vice vera, e.g. address
  - Mapping relationship due to difference in data structure:
    - One-to-one
    - One-to-many
    - Many-to-one
    - Many-to-many

IX. Conversion Plan

A data conversion needs to be drafted with the following in details:

- List of data in source and their volume
- Data Mapping and Data Transformation rules
- Overall Approach: Rounds of Trial Run and Rehearsal, Freeze Period, Cut-over day
- Extraction, conversion and verification program logic
- Data Verification and Acceptance rules
- Result Report
- Handling of obsolete data

Chatper 5. Security & Privacy Protection

I. Definition

Security refers to protection of computer system and the information under processing against unauthorized and deliberately access, modification or destruction, especially on confidential, sensitive or personal information. Protection of the later refers to privacy control, usually govern by legal regulation on the collection, usage, storage and distribution of the personal data.

II. Identify Threat

Security Thereat could be grouped into the following types:

I. Data Leakage
- ◆ Confidential information displayed without encryption
- ◆ Unauthorized access to read specific data
- ◆ Data transfer to other persons or location, either unauthorized or unintentional
- ◆ Physical loss of data stored in electronic media such as USB flash disk

II. Hacking

Break into computer system for the following purpose:
- ◆ Access or steal data
- ◆ Modification of transaction data
- ◆ Modification of information display, i.e. web defacement
- ◆ Modification of information transmit, i.e. alter message
- ◆ Record deletion e.g. log
- ◆ Take control of the computer system, e.g. enable or disable door lock

III. Malfunctioning
- ◆ Disable specific computer system or devices from service
- ◆ Encryption of data to prevent enterprise from normal operation
- ◆ Replacement of specific computer system or devices

III. Identify Attack

- Malware
  1. Virus
     Computer program that could corrupt or delete data on computer.
  2. Trojans
     Program download onto a computer that disguised as a normal program that cause harm.
  3. Spyware
     Software that enters a computer, gathers data and sends it to third parties without consent.
  4. Worm
     Computer program that utilizes security loophole (vulnerabilities) of a target computer to replicate itself and spread over the computer network so as to inflect more and more computers. It will consume large memory and bandwidth.
  5. Keyloggers
     Software or hardware that records keystrokes type so as to steal password or other confidential information.
- Denial-of-Service (DDoS)
  Make server machine or network resource to become unavailable by single or multiple system flooding using excessive requests on them.
- Zero Day Attack
  Unreported or announced software vulnerability that hackers can exploit and leave zero days to create patches or have workarounds.
- Advanced Persistent Threat
  Continuous hacking processes over long period of time that exploit vulnerabilities in systems using malware, web phishing or email with malicious URLs, in order to monitor and extract data from a specific target, to collect information on surrounding infrastructure, to crack password and acquire administrator privileges, to expand control to other workstations and servers.
- Ransome
  Lock user files, applications or systems by encryption of computer's master file table or entire hard drive until a ransom is paid usually through a hidden ransomware link in email or web page that could establish a connection to callback server that set up unique keys to encrypt the data on target machine

IV. Implement Prevention Measures

Data

- Identify and classify information that need protection:
  secret, confidential, sensitive or other important data such as personal identification information
- Encrypt personal identification information in print media, report, log, database, etc.
- Encrypt password in log, database, data or configuration file, etc.
- Maintain and update access control list on data by user group

System Application

- Maintain and update access control list on data and function by user group
- Apply hash to personal identification information (print, report, log, database, etc.)
- Personal password or identification information should be hashed and cannot decode
- System password should be encrypted or masked (log, database, data or configuration file, etc.)
- Encryption apply to data storage or transmission
- Key length of symmetric encryption such as AES should be at least 128-bit while asymmetric encryption such as RSA should be at least 2048-bit
- Cryptographic key for encryption & decryption should be changed from time to time
- Adopt 2-factor authentication (SMS/email) or challenge-response scheme for important service or transaction. User have to provide correct response before log in, such as CAPTCHA (Completely Automated Public Turing test to tell Computers & Humans Apart) for form input submission

V. Common Vulnerabilities

i. Broken Authentication and Session Management

Compromise poorly protected passwords, keys, or tokens to act on other user identities.

ii. Cross Site Request Forgery (CSRF)

Force logged-on browser to send pre-authenticated request to vulnerable web application so as to allow browser to perform hostile action

iii. Cross Site Scripting (XSS)

Execute script in browser to hijack user sessions, deface web sites or even introduce worms

VI. Establish Incident Handling Procedures

- Follow existing plan & procedures, e.g. Incident Response, Data Backup & Recovery or Operation
- Reporting
  - Report the case to senior & responsible staff
  - Inform administrator
  - Notify end users best with estimated service resume time
- Immediate actions to limit the impact:
  - Disable public access for web defacement by offline web server
  - Disconnect network cable to avoid affecting other computers
  - Power off computer to stop ransomware from encrypting more files

- Server operation
  - Switch to alternate sites or networks if possible
  - Perform disaster recovery within required time frame and start up the resilient system
- Trace and Fix Problem
  - Trace root cause of the threat by referring to server log and system event
  - Check what data, file and programs, URL or email are accessed before the incident
  - Trace attacker source IP addresses or other information
  - Trace how the attack is carried out and what changes were made

- Further Follow up Action
  - Report to the Police for investigation
  - Change all passwords involved
  - Updated server patches and further change configuration or setting if necessary
  - Update manual, e.g. Incident Response Plan, Data Backup & Recovery Plan or Operation Manual procedures with following details:
    - ◆ Simple and clear steps to follow, including how to check and what to change

VII. Enforce Privacy Protection

Storage and Transmission

- Personal identified information should be hashed to store and transmit in electronic form.
- Non identification personal information to be encrypted in storage and during transmission.
- Data transmission should be in secure channel.
- For two information that could identify a particular person, one of them should be encrypted.

Printout and Copy

- No unauthorized printout or copy of personal information
- Authorized printout or copy of personal information should be kept and covered

- No private storage of personal information in any form

Data Transfer

- Transfer of personal information in electronic form should be encrypted, and put in sealed envelope, bag or locker box for physical form to prevent data leakage to unauthorized parties.
- Transfer of original copy of personal information depends only on reliable staff, transport or couriers with sufficient packaging to prevent physical or damage.

Data Collection

- Only partial personal identification information could be obtained for normal business verification need.

Testing and Support

- Production data if used for testing purposes, should be masked or encrypted, and they should be cleared immediately after testing.

Data Retention

- After authorized or legal retention period, data should be disposed in a non-recoverable and not human understandable format. Data that need to be kept

V. Invitation for Quotation or Tender

-Collect a contact list of suppliers with company name, staff name, title, email and telephone.
Usually have dedicated person to follow account of large corporation or public organization
-Public Organization may consider only a supplier in pool that require certain condition to join
-Invitation for quotation or tender could be sent out by email. Tender requirement,

VI. Evaluation & Award

- ■ Evaluation is based on agreed rules and treatment should be equal to all supplier
- ■ Grading is calculated according to marking scheme
- ■ With same functions and features provided, if performance is also the same, award should be given to the supplier or bidder with lowest product or service price

VII. Purchase Order

Purchase order need to prepare and send to vendor after award with the following information:

- ■ Order details on goods or services to be purchased, include brand, model name and quantity
- ■ Contact Information of the buyer and vendor
- ■ Shipping and Billing Address
- ■ Contract and Payment Terms, including price, discounts or sales tax

Chatper 7. User Acceptance Test Plan

I. Purpose

1. Test if the system meets the followings:

iii. User requirement specified in requirement, system or functional specification

iv. User expectation regarding system operation and user interface

2. Identify all specific and potential issues, correct and clear them before production

II. Deliverables

1. **UAT Plan**
2. **Test Case**

-Drafted by development team, amended and confirmed by users
-To be filled out by testers during User Acceptance Test
-Either in word or excel format
-Include the following items:

v. Test Case Number
vi. Test Scenario Description
vii. Operation Steps
viii. Expected Result

VII. Issue Logging

Tester needs to record down issues found during UAT, either in excel or in logging system such as Redmine with the following information:

1. **Type**
   - **Bug**
     Function not performed as expected or specified
   - **User Interface**
     Navigation, text display or graphics issue
   - **Performance**
     Loading and response time
   - **Setup**
     Incorrect system configuration or settings

2. **Classification**

   i. **Severity**
   - **Critical**
     System function unable to execute
   - **High**
     Bug found or function different from expectation

   ii. **Urgency**
   - **High**
     Need to handle immediately
   - **Medium**
     Need to handle within a fixed period

XI. Administrative Procedure

- Attendance signature may be required for outside testers

XII. System Shutdown & Resume

- After bugs or issues are fixed, system need to shutdown to deploy the amended application
- Development team will send email to inform users before the deployment and after the system is resumed.

XIII. Review Meeting

- After each round of UAT is completed, review meeting should be held between development team and user whether UAT is passed or not, can continue the next step or what to follow.

XIV. Quality Factor

Common factor regarding user requirement that could affect the UAT performance leading to a high failure rate or many issues found:

- Not documented clearly
- Incorrect documentation
- No in depth understanding
- Misunderstanding
- Different interpretation

III. Rollout Planning

- Identify **major & supplementary tasks**
- Check **task dependency**, and **prioritize** them
- **Estimate resource** required
- Identify **stakeholder and parties** responsible for each task
- List out **acceptance criteria**
- Establish **measurable targets**

IV. Preparation Task

1. Clarify Contact Point, Roles & Responsibility

ROLE	RESPONSIBILITY
User	5. Provide business requirement 6. Carry out result verification 7. Approval and acceptance
Project Team	2. Monitor & review whole process 3. Liaison with User & Technical Teams 4. Vendor Management
Development Team	6. Functional and technical design 7. Code development and bug fixing 8. System setup and configuration 9. Application Deployment 10. Extract, transform and load data 11. Data cleansing 12. Documentation
Infra & Network Team	4. Server and cloud environment preparation and setup 5. Network cabling 6. Firewall and proxy configuration 7. Documentation

4.Rollout Steps

- Old system stops and new system startup steps should be listed out.
- Old system stop sequence should be as follows and each may have more than one instance:
  - Web Server, Application, Database
- Execute Data Conversion Script to extract, transform and load production data to new system
- Wait for other system to rollout if there are inter-system dependance
- Collect conversion log and compile exceptional report
- Provide rollout status, data conversion success and failure records, list out error log or other problems encountered, suggest solutions of handling obsoleted or incorrect data
- User Verification

5.Fallback Steps

- Condition needs to be agreed under what situation the old system should be restored, i.e. not pass health check and serious problem discovered.
- Some minor issues such as incorrect text displayed could be fixed immediately not affecting rollout
- New system stop sequence should be as follows and each may have more than one instance:
  Web Server, Application, Database
- Old system startup sequence should be as follows and each may have more than one instance:

2. Interactivity

Tasks	Start Day	Finish Day
Requirement Collection	**DD/MM/YY**	**DD/MM/YY**
Requirement Specification		
Product Analysis	**DD/MM/YY**	**DD/MM/YY**
Functional Specification		
Interactivity Design	**DD/MM/YY**	**DD/MM/YY**
Information Design		
Interface Design		
Interaction Design		
Prototyping	**DD/MM/YY**	**DD/MM/YY**
User Acceptance Test	**DD/MM/YY**	**DD/MM/YY**
Production Rollout	**DD/MM/YY**	**DD/MM/YY**
Maintenance Review	**DD/MM/YY**	**DD/MM/YY**

3. Architecture and Coding

Tasks	Start Day	Finish Day
Requirement Collection	**DD/MM/YY**	**DD/MM/YY**
System Analysis	**DD/MM/YY**	**DD/MM/YY**
System Design	**DD/MM/YY**	**DD/MM/YY**
Documentation	**DD/MM/YY**	**DD/MM/YY**
Platform Setup	**DD/MM/YY**	**DD/MM/YY**
Implementation	**DD/MM/YY**	**DD/MM/YY**
Coding		
Unit Test		
Installation and Deployment	**DD/MM/YY**	**DD/MM/YY**
Data Migration & Conversion	**DD/MM/YY**	**DD/MM/YY**
Testing	**DD/MM/YY**	**DD/MM/YY**
SIT		
UAT		
Security Audit & Privacy Review	**DD/MM/YY**	**DD/MM/YY**
System Rollout	**DD/MM/YY**	**DD/MM/YY**
Project Review Report	**DD/MM/YY**	**DD/MM/YY**

Reference

1. Project Planning

1. 101 Common Causes - Why Do Projects Fail? International Project Leadership Academy, calleam.com, 2016
2. 25 Point Implementation Plan to Reform Federal Information Technology Management, Vivek Kundra, The White House, 2010
3. A Guide to Project Management Body of Knowledge (PMBOK Guide), 6th Edition, Project Management Institute, 2017
4. A Project Manager's Book Of Forms, A Companion To The Pmbok Guide(Sixth Edition), 3rd Edition, Cynthia Snyder Dionisio, Wiley,2017
5. Agile Coaching, Rachel Davies and Liz Sedley, Pragmatic Bookshelf, 2009-10
6. Agile Practice Guide, Project Management Institute, 2018-11
7. Agile Principles, Patterns, and Practices in C#, Robert C. Martin Micah Martin, Pearson, 2006-07
8. Designing Visual Interfaces: Communication Oriented Techniques, Kevin Mullet and Darrell Sano, Prentice Hall, 1994
9. Exploring Requirements: Quality Before Design, Donald C. Gause and Gerald M. Weinberg, Dorset House, 1989-09
10. Head First Design Patterns, Eric Freeman, Bert Bates, Kathy Sierra and Elisabeth Robson, O'Reilly Media, 2004
11. Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project, Tom Kendrick, AMACOM, 2009
12. Improvement Measures for Delivery of IT Projects, OGCIO, 2014
13. Information Technology Project Management, Kathy Schwalbe, Thomson Learning, 2002
14. IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices, Ryan Nelson, Amazon, 2020
15. IT 真相 打通 IT 與商務的通路, 楠真著, 廣東人民出版社 , 2019-01
16. Major Causes of Software Project Failures, Lorin J. May, Department of Computer Science, University of Brasilia
17. Management Capability Assessment, Dennis Wong, Bleu Publications, 2021
18. Microsoft .NET-Architecting Applications for the Enterprise, Dino Esposito & Andrea Saltarello, Microsoft Press, 2014
19. Multimedia ISO 9000 Logic Handbook, China Orient QA Co., Limited, 1994

3. Architecture and Coding

System Design

- 12 Essential Skills for Software Architects, Dave Hendricksen, Pearson, 2012
- 12 More Essential Skills for Software Architects, Dave Hendricksen, Pearson, 2015
- The Art of Scalability, Martin L. Abbott, Addison-Wesley Professional, 2015
- Clean Architecture: A Craftsman's Guide to Software Structure and Design, Robert C. Martin, Prentice Hall, 2017
- Designing for Behavior Change, Stephen Wendel, O'Reilly, 2013
- Enterprise Integration Patterns, Gregor Hohpe & Bobby Woolf, Pearson, 2003

- SOA Design Patterns, Thomas Rischbeck & Thomas Erl, Prentice Hall, 2009
- Strategy & Product Development for Complex Systems, Edward Crawley, Bruce Cameron & Daniel Selva, Pearson, 2015
- 軟件架構設計, 温昱, 電子工業出版社, 2012-10
- 構建高性能 Web 站點 改善性能和擴展規模的具體做法, 郭欣, 電子工業出版社, 2009-08
- 億級流量系統架構設計與實戰, 李琛軒, 電子工業出版社, 2024-05
- 億級流量網站架構核心技術, 張開濤, 電子工業出版社, 2017-04
- IT アーキテクト最強の指南書, 日経 IT エンジニアスクール, 日経 NETWORK, 日経 BP, 2016-06

System Development

- Algorithms, 4th Edition, Robert Sedgewick and Kevin Wayne, Addison-Wesley Professional, 2011-03
- Effective Debugging, Diomidis Spinellis, Pearson Education, 2017
- Effective Java, Second Edition, Joshua Bloch, Pearson Education, 2008
- Effective JavaScript, David Herman, Pearson Education, 2013
- Guide to the Certified Software Quality Analyst (CSQA) Common Body of Knowledge
- Head First EJB, Kathy Sierra, Bert Bates, O'Reilly, 2003
- Head Rush Ajax, Brett McLaughlin, O'Reilly, 2006

- Software Exorcism, Bill Blunden, Apress, 2003
- Software Requirement Patterns, Developer Best Practices, Stephen Withall, Microsoft Press, 2007-06
- 今晚來點 web 前端效能優化大補帖, 莫力全 Kyle Mo, 博碩, 2022
- 你所不知道的必學前端 Debug 技巧, 楊楚玄, 博碩, 2021
- Web 開発者のための大規模サービス技術入門, 伊藤 直也/田中 慎司, 技術評論社, 2010-07

Database & Performance Tuning

- Effective MySQL Optimizing SQL Statements, Ronald Bradford, 2018
- Effective SQL, John L.Viescas, Addison-Wesley, 2017
- High Performance MySQL, Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, 2012
- Oracle Database Problem Solving and Troubleshooting Handbook, Tariq Farooq (etc.), Addison-Wesley, 2016-04
- Oracle High Performance SQL Tuning, Donald Burleson, Oracle, 2001
- Scalability Rules: Principles for Scaling Web Sites, Martin Abbott, Michael Fisher, Pearson, 2017
- 品悟性能優化, 羅敏, 清華大學出版社, 2011-05
- 海量資料庫解決方案, (韓)李華植, 電子工業出版, 2013-05

System Security

- Java Security Handbook, Jamie Jaworski & Paul Perrone, SAMS, 2000
- Penetration Testing: A Hands-On Introduction to Hacking Penetration Testing, Georgia Weidman, No Starch, 2014-06
- SQL Injection Attacks and Defense, Justin Clarke, Kevvie Fowler, Syngress, 2009
- XSS Attacks: Cross Site Scripting Exploits and Defense, Seth Fogie, Jeremiah Grossman, 2007
- セキュリティ最強の指南書, 日経 IT エンジニアスクール, 日経 NETWORK, 日経 BP, 2016-07

User Interface

- Designing Business: Multiple Media, Multiple Disciplines, Clement Mok, 1996
- Design for How People Think, Stephen Wendel, O'Reilly, 2019-04
- Designing with the Mind in Mind, Jeff Johnson, Elsevier, 2014

Internationalization

- CJKV Information Processing, 2nd Edition, Ken Lunde, O'Reilly, 2009-02
- 中文系統徹底研究 輸入法與秀字, 李明清, 旗標, 1992
- 中文數位探索：從漢字輸入到電腦中文化的壯闊歷程, 墨磊寧, 台灣商務, 2025-02
- 遨遊 C 語言中文顯示的製作, 高衡緒, 和碩科技, 1993-08
- ゼロからの OS 自作入門 , 内田公太, マイナビ出版, 2021-03
- 日本語入力を支える技術, 德永拓之, 技術評論社, 2012-03

Geographic Information System

- Building a GIS, Dave Peters, ESRI, 2008
- Building Web and Mobile ArcGIS Server Applications with JavaScript, Eric Pimpler, Packt, 2014
- Designing Geodatabases, David Arctur & Michael Zeiler, ESRI, 2004

About the Author

Nicho Wong has over 25 years of working experience in the information technology industry.

As a development manager, system analyst, analytical programmer and associate producer for many years, he has been involved in many different kinds of projects, including portal site, e-commerce website, content management system, stock information system, electronic support system, geographic information system, e-learning system, multimedia educational CD-ROM titles, mobile and social media games, and many other government systems.

He has worked for 16 departments of the Hong Kong and Singapore government, 2 major telecommunication companies, banks, airline, and advertising company, including small companies, famous public listing companies and multinationals.

Project size varies from small, medium to large. Team size ranges from two to hundred people. Project expenses incurred range from hundred thousand to million, and even hundred million of Hong Kong dollars.

He was graduated from the Singapore Polytechnic with Advanced Diploma in Multimedia Development and from the University of Greenwich with a Bachelor of Science degree in Computing.

He could be contacted by nichowong@dataflyer.net.