University of Cyprus
Department of Electrical and Computer Engineering

Development of Database System for Publications and Thesis

FROM
CHRISTOS KATOMONIATIS

Submitted to the University of Cyprus as partial completion of the requirements for a degree in Electrical and Computer Engineering

Department of Electrical and Computer Engineering
May 2017

Examination committee
Michael Maria
Professor, Department of ECE, Advisor

ACKNOWLEDGEMENTS

After so long of hard work and I can not believe it is finally finished!
All those people who contributed in my work during that period I would like to thank them all.

First and foremost I would like to thank and express my sincere gratitude and unrestrained appreciation to my supervisor Dr. Maria K. Michael, for her continuous support, patience, motivation and immense knowledge, and providing me with an excellent atmosphere for doing my project work. You are a great professor, advisor and supporter and I really appreciate every second you dedicated to our meetings in order to complete this project.

Furthermore, I would like to thank some of my friends and colleagues who helped me a lot during my thesis. I would like to thank Savvas for providing me a lot of help regarding Pages and also for formatting this report. I would like to thank my friend Marios who helped me in reviewing English language.

Beyond friends, there is family. Thanks for your love and support for no matter what I decided to do, you were always there for me.

Christos Katomoniatis
BSc Student
University of Cyprus,Cyprus
Email: ckatom01@ucy.ac.cy

ABSTRACT

Technology evolves in the department of computers which leads to the increase of data we have to manage. Thus creating the need of systems that can manage big data in fast and effective ways. In the long run, saving loads of data by creating new systems becomes a necessity for an organisation or an enterprise.

Needles to say that the creation of a system could benefit in the management of data, which also gives the opportunity to read, write, edit, update, delete, search in a database through user friendly interfaces. With a proper design and implementation of database , the data could be fetched easily and fast enough with multiple queries and simultaneously give the ability to modify the data.

In this thesis it is presented the analysis, design and implementation of a system where publications and thesis are going to be posted through a website. Before the implementation of this system, there was no way to archive, manage and report all those types of publications of the faculty and researchers of the ECE department as well as various Diploma, MSc and PhD Theses.

Although the primary purpose was not just to implement this system, but to build a system with correct and understandable code that changes could not affect the design of the database. The implementation of this system ensures the security, access and management of data for authors who submit their publications and gives them the opportunity to create reports of their own. The system was implemented in PHP, HTML, CSS for the user interface, Javascript for the back-end functions and SQL (Structure Query Language) that was used for the creation and design of the database.

Contents at a Glance

Chapter 1

1.Introduction

"*The beginning is the most important part of the work*"

-Plato-

1.1 General

There are several challenges in today's technological systems and organising and storing data is one of them. Without a doubt the quantities of data now available compared to the quantities of data of the future are exiguous. But what is the key requirements for big data storage? Obviously, how well it can handle those very large amounts of data, performance, security with build, robust quality and well-maintained design. In design and documentation if conflict arises, clarity should be preferred to precision because, as will be described, the key problem of software development is having a functioning communication between the involved human parties. The developer should always keep in mind that the software is written for people and not for computers. Computers just run this software which is a minor point in my perspective. It is people who understand, maintain, improve and use the software to solve the problems that might occur.

Software engineering is solving a problem by an effective abstraction and representation. The existing technologies evolve or become obsolete, but the underlying principles and concepts will likely resurface in new technologies. A single developer might not be able to solve a complex problem. With software engineering, is not consisted with only a developer but also with quality assurance tester, system architect, platform engineer, costumer, project manager. Writing the code is not just the implementation, it involves the guidelines and writing documentation and also writing unit tests. Those unit tests should fit together in order to spot the problematic areas using metrics and furthermore improve the quality there.

1.2 Purpose

In this thesis is presented the analysis, design and implementation of an information Web-based management system , which ensures security, access and management of data for authors/users and admins. Before the implementation of this system, publications and theses were a sea of unorganised data which took time and effort to find, handle, store or even search. Therefore this system would be exactly what is needed in order to daily monitor and process loads of data. More specifically the system can add, edit, update, delete, import, export, search and fetch data throughout 5 tables that are stored in MySQL database. The system has been implemented in PHP,HTML,Javascript Languages, Web services, Apache Server for the website. Additionally the MySQL Language (Structure Query Language) used to create the databases.

1.3 Vision

What is required in this system is speed, simplicity and security. Envision being able to search for a particular author or journal or even a date and edit or update the database within a click. Computer technology through the years has greatly introduced the idea of automation and storing large data as well as being a learning tool for those who seek it. Nowadays everyone has access to the internet in some way, thus to allow for effortless accessibility to the common use, requiring a web based software service.

1.4 Glossary of Terms

Database: an organised collection of data that are typically organised to model relevant aspects of reality in a way that supports process rewiring this information.

Algorithms: a step-by step procedure for calculations.

Preview List: a list of tracked contracts or projects stored within the system for a particular registered user.

Web service: is a method of communicating between two electronic devices over the World Wide Web.

SQL: is a computer language databases, designed for data management in a management system relational database (RelationalDatabaseManagementSystem, RDBMS), which initially was based on relational algebra. The language includes features retrieval and data update, shapes creation and modification and relational tables, and access control to the data.

SQL Server: It is a relational database management system (RDBMS) from Microsoft that is designed for the business environment.

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

Requirements analysis: (requirements engineering) is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

Javascript: is an object-oriented computer programming language commonly used to create interactive effects within web browsers.

1.5 Requirements Analysis

Identifier	Points*	Requirements
REQ1	4	The system shall ensure that all user password shall remain secure
REQ2	3	The system should have loading times no greater than 3 seconds with fair broadband speeds, great performance
REQ3	5	The system should provide the user/admin with options in the menu
REQ4	5	The system can support multiple users and a few admins
REQ5	7	The system should be able to fetch data with a click of a button
REQ6	10	The system should be able to add, delete, edit or update the data in forms
REQ7	5	The system must preview all the data in csv format
REQ8	5	They system should be able to import and export files in comma separated values (also known as csv) format
REQ9	5	The system can print and export in pdf/csv searched data in the Compile Report
REQ10	15	The system can search specific data in the database (search as authors, year and tables) with multiple sql queries
REQ11	5	The system can present reports of searched data in the Compile Report
REQ12	3	The system should validate the added forms
REQ13	5	The system should verify the users/admin by User ID on log in
REQ14	10	The system can verify the user's ID in order to allow to add or import any data in the website

Identifier	Points*	Requirements
REQ15	15	The system allows the administrator to manage the whole system
REQ16	5	They system can create a Compile Report where all the tables can be searched
REQ17	2	The system should be connected to a MySQL database
REQ18	15	They system should use a web server (Apache), MySQL as the relational database management system and PHP as the object-oriented scripting language
REQ19	5	The system should be accessible only on the network of the KIOS Research Centre and will be restricted at public networks
REQ20	5	The system should use Javascript for the back-end functions of the website

Points* = the higher the number of points refer to the higher difficulty of the task
REQ = stands for the **requirements** that the system needs to have in order to fully accomplish the manager's expectations.
So this lead us to the complexity of the project, also known as Technical Complexity Factor, which a table below previews the non-functional Requirements.

Technical Complexity Factor (TCF) — Nonfunctional Requirements

Technical Factor	Description	Perceived Complexity	Weight	Calculated Factor
T1	Distributed system (running on multiple machines)	3	2	2x3=3
T2	Performance objectives	3	1	1x3=3
T3	End-user efficiency	3	1	1x3=3
T4	Complex internal processing	3	1	1x3=3
T5	Reusable design or code	0	1	1x0=0
T6	Easy to install	3	0.5	0.5x3=1.5
T7	Easy to use	5	0.5	0.5x5=2.5

Technical Factor	Description	Perceived Complexity	Weight	Calculated Factor
T8	Portable	2	2	2x2=4
T9	Easy to change	1	1	1x1=1
T10	Concurrent use (by multiple users)	4	1	1x4=4
T11	Special security features	5	1	1x5=5
T12	Provides direct access for third parties(the system will be used from different organisations)	1	1	1x1=1
T13	Special user training faculties are required	0	1	1x0=0

The TCF is one of the factors applied to the estimated size of the software in order to account for technical considerations of the system. It is determined by assigning a score between 0 (factor is irrelevant) and 5 (factor is essential) to each of the 13 technical factors listed in the table below. This score is then multiplied by the defined weighted value for each factor. The total of all calculated values is the technical factor (TF). The TF is then used to compute the TCF with the following formula =>

Column1=C1 = 33
C2= 14
C3 = TFT = 34
Result of Technical Complexity Factor (TCF) => TFC Formula =>
TCF = 0,6 + (TF/100) => **TF = 34**, from above Table - Column 3
TCF = 0,6 + (34/100)
TCF = 0,6 + 0,34
TCF = 0,94

When applying any general cost estimation technique, you have to account for many variables. Every software project is different, and if you don't account for those differences, your estimation will not be reliable.
All factors do not have the same potential impact on a project cost estimate, so each factor has a multiplier, representing the relative weights of the factors.

Chapter 2 - Object Oriented Programming Concepts

2.1 Object Oriented Analysis

Object oriented programming methodology based on objects which is concerned to develop an application, a web-based system, on real time, so more emphasis is given on data unlike the other programming styles like structured or functional. The concept of having objects makes OOP more organised, reusable and speedy. Today most of the programming languages support OOP. Most of them have enhanced features to make OOP more easy and maintainable.

So firstly let's analyse an object. An object consists of attributes and methods. Attributes define the properties of the objet while methods define its behaviour. In OOP, object is considered as an instance of a class. Considering our web-based system , "Admins" and "Users" can be considered as objects and "Users name" can be considered as their attributes.

Secondly, a class is a simple representation of a type of object. Using the blueprint analogy, a class is a blueprint of an object and an object is a building made from that blueprint. It consists of a name , attributes and methods. Encapsulation , inheritance and polymorphism are the main fundamental concepts in OOP.

Inheritance

Inheritance describes the ability to create new classes based on an existing class. So the object can inherit properties from another object while defining common code only one place with 1 enhancing maintainability of the software. There are many examples that could represent what inheritance means. So for instance we have a "system", which is the parent class. And it is connected with "database" and "website" below the system. Both "database" and "website" inherit all the fields and methods of the parent class "system" . Hence inheritance guarantees code reusability.

Polymorphism

Polymorphism is the ability to take more than one forms depending on data type or class, so that multiple classes can be used interchangeably, even though each class implements the same properties or methods in different ways. Hence polymorphism guarantees maintainability of the software. Interface introduces polymorphism. It contains only definition of methods, properties and events, so the class that implements the interface, has the implementation or declaration to achieve polymorphism.

Advantages of Object Oriented Programming

• Easy to maintain and extend existing code

• Enhanced code reusability
• Object hiding can be achieved
• Improved reliability and flexibility, as objects can be dynamically called and accessed, new objects may be created at any time
• Faster development, as reusing software modules lowers the time usage
• Cost effectiveness, as reusing software modules lowers the cost of development

Disadvantages of Object Oriented Programming
• Need extra time and effort for planning
• Not suitable for all types of problems
• OOP programs are generally larger and slower than normal programs

Web services

Web service is a software system designed to connect to other software applications. An application can publish its function or message to the rest of the world using web services. Web services use XML to code/decode data. Other applications can interact with web service using SOAP messages.
Web services offer many benefits over other types of such systems.
• Interoperability – This is the most important benefit of web services. Web Services allow different applications to interact with each other and share data and services among themselves. For example VB or .NET application can communicate with java web services and vice versa.
• Low cost communication as web services use SOAP over HTTP protocol.
• Reusability
• Usability – Other applications have the freedom to choose the web service they need.

A logical view of web services architecture which is based on three primary roles, service requester, service provider and service registry. Providers of the web services are known as service providers, users of the web services are known as clients or service requester. The essential part of web services is the interact relationship between a service provider and a service requester. These three roles interact using publish, find and bind operations. The service provider publishes the service description in a service registry. The service requester finds the service description in a service registry and uses the information in the description to bind to a service. The service registry provides a centralised location for storing service descriptions. They will become interface to a huge world of data and query services. UDDI, a platform independent, XML based registry is an example of service registry.

2.2 Software Development Process

Software Development Process commonly comprises sequence of work activities, actions, and tasks that are undergone to create the final product. In the context of software

development, the final product is a software application (the web-based system), a plug-in component, or a software service solution. However, software development processes are complex and unmanaged process could easily lead to catastrophic failure in delivering a usable system. While there are many factors that contribute to its complexity, the two main reasons described by:

1) Intellectual and creative processes rely on people's decisions and judgement and
2) The environment may vary hence producing rapidly changing software requirements or strictly defined criteria.

Consequently, careful planning of development activities is required and this results in the adoption of software development process model. A software process model is an abstract representation of interrelated activities in software development . It describes the general approaches in structuring activities and some techniques to produce deliverables. Selecting a suitable model would cut down the development time and increase the quality of the output. Following sections describes several widely applied software process model in the software industry.

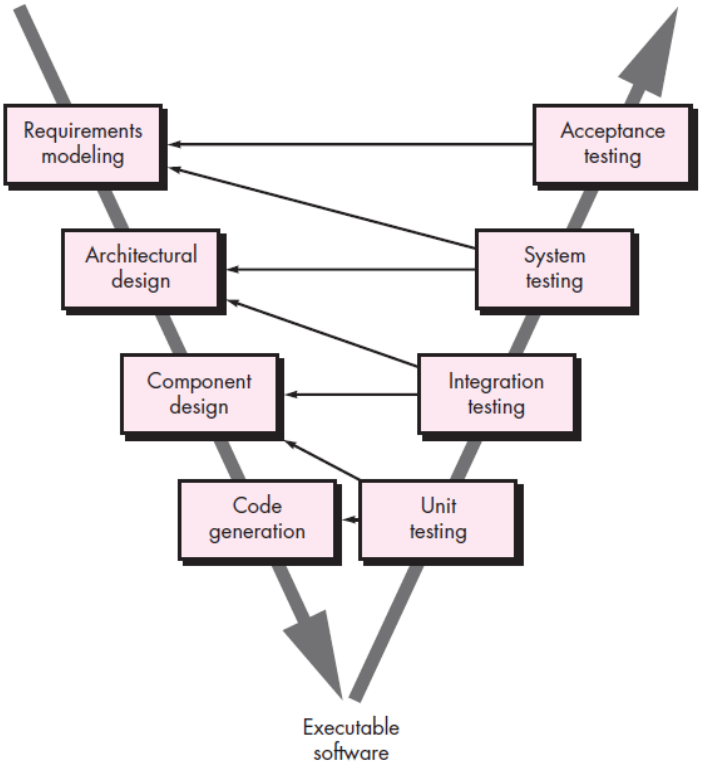
Waterfall Model

Waterfall model is a traditional software process model introduced by Royce. It is a rigid and linear document driven methodology. This model is known as the waterfall model because it proceeded from one phase to another in a cascading order. Before each phase can begin, each of the phases has a definite set of deliverables that must be approved by project sponsor or project supervisor, after the stakeholders have elicited them. However, the process of producing and approving these deliverables will incur significant cost. The waterfall model often receives criticism on its inability to accommodate changes because the project freezes system specification upon deliverables sign-off. In a dynamic business environment, it is often difficult for user to state all requirements explicitly. The waterfall model lacks the ability to accommodate natural uncertainty and the changing need of users.

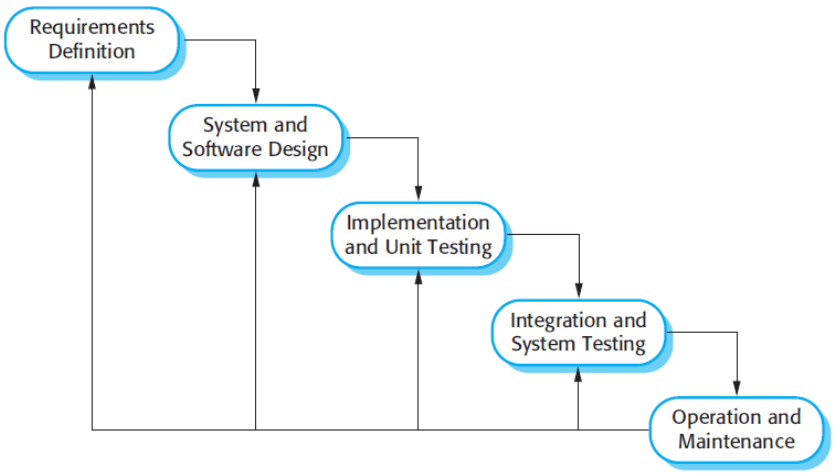
Overview of the waterfall model

Another serious disadvantage of the waterfall model is that testing is often left to the end of the project. Errors and feedbacks obtained in later stages will require additional effort to resolve. Eventually, this will lead to a software product that not fit for user need. An enhanced variant of the waterfall model known as the V-model has improved to this issue.

Figure at the left illustrates the quality assurance actions associated with deliverables of earlier phases in the V-model. Verification and validation approaches applied to earlier



engineering work could significantly reduce errors found in later stages. However, the V-model does not explicitly describe actions taken in order to deal with errors found during testing.Nevertheless, waterfall model does show its strength when used in project where requirements are well understood and stable during development . Documents produced during each phase provide traceability to address safety and legal issues when such concerns are critical to the user.



Evolutionary Model

Evolutionary model encapsulates two fundamental approaches: incremental and interactive. When addressing changing requirements. It organises processes in a manner

that enables the development of increasingly complete versions of software based on customer feedbacks through a series of iteration. The two fundamental types of evolutionary model that will be covered are prototyping and spiral model.

The idea of prototyping is to enable users to interact and experiment with early prototypes which encapsulate a set of mutual understanding requirements. In [22], prototypes are described as an initial version of a software system used to demonstrate concepts, explore design options, in-depth problems and their possible solutions. Commonly, there are two types of prototypes:

a.**Common prototypes** aim to explore customer requirements through building an incrementally usable system. Prototypes with a minimum set of basic requirements are built and presented for the customer's evaluation. The prototypes evolves by the implementation of customer proposed features and changes until it's functionalities finally agreed by customer

b.**Throwaway prototypes** aim to gather information and generate ideas on how system should be built. Commonly during project start up, the user may not fully understand their need and the developer may not share understanding on certain features. To clarify these uncertainties, a design prototype which contain just enough details is built for evaluation. Once issues have been clarified, developers could then move on to an actual design and implementation.

Allowing requirements to be implemented rapidly is the key advantage to prototyping. However, this may lead to stakeholder confusion by mistreating what they see as final version of the system. Stakeholders should be well aware that some prototypes only serve as tools to gather requirements and may vary from the final product.

The spiral model is an evolutionary process model that combines the iterative nature of prototyping while retaining the systematic approaches of waterfall model which was explained earlier. Unlike the waterfall model in which it is hard to backtrack to previous phases once deliverables freeze, spiral model can be adopted throughout system development phases which takes its name from spiral (the widening curve). Explicit recognition of risk in the spiral model is the main difference compared with other process models . However, understanding and mitigating the risks potentially reduces things that can go wrong. Iteration over entire phases of software process would be costly. Hence, the spiral model is more suitable for large-scale projects, which contains high risk and requires well-structured approaches.

Although for projects like web-based database system the spiral model is not very suitable but it definitely requires the same well-strutted approaches.

Agile Development

Agile development processes have emerged to be the dominant software process model in recent years. Agile processes focuses on people, communication, working software, systems and responding to change as opposed to plan-driven models that have high process bureaucracy. Design and implementation are the central activities in agile development process. It would also be possible to incorporate requirements elicitation and testing into these activities, for instance, applying test-driven development (TDD). In TDD, the developer first writes test cases before writing actual implementations. This serves as the preliminary steps to clarify requirements and understanding for problem domains. Developers then code the actual implementations and execute tests to verify the implementations.

Furthermore, a comparison of the three models discussed above based on several concerns that may affect development activities. These concerns, together with their explanations, are listed below:

a. Requirement elicitation, presents approaches to gather requirements for system

b. Change management, reflects how changes will be handled throughout project

c. Validation, explains when testing will be done during project

d. Delivery discuss how quickly and often the software features will be delivered and

e. Design modelling covers the depth of design processes during modelling activity.

Based on the comparison, agile development clearly exhibited features that meet the needs. Requirements of user tasks and roles to model intuitive UI. Apparently, user stories of agile development fit better with these requirements. In addition, agile development has factored change management in the model. Its ability to cope with changes reduces the risk of delivering products that does not meet the objectives. The earlier the system is tested, the less effort will be spent on the error that may arise in end of the project. TDD of agile practise embraces this idea and encourages testing done before development. Connected to this, frequent delivery also implies that new enhancement have actually been verified in smaller scale. It reduces complexity by testing only parts that have been changed. Upfront design often leads to design paralysis when the developer tries to adopt concerns and considerations that may not be materialised in the future of project. This is why agile development prefers modelling just enough detail to support an anti-pattern that involves excessive up-front analysis and design but no actual action taken and refactor as required. Finally, lightweight agile process fits well into small-scale development which involves only single developer.

2.3 Requirement Engineering

The techniques utilised to gather and analyse requirements are discussed which then leads to documents on functional and non-functional requirements. These requirements are then translated into informative models to assist decision-making and problem understanding in the Requirement Modelling phase. Finally, the chapter reviews several techniques to manage software development activities that could help to realise these requirements.

The first step of the project is to understand user (and indeed stakeholders) requirements for building this system. Requirements of a system can be defined as descriptions of what services it could provide and the constraints on its operation . These requirements directly address user needs in term of using the system to achieve their business operation – or process.

The process of understanding requirements, known as Requirement Engineering, involves a broad spectrum of processes, tasks and techniques to collect, analyse, document, and verify the requirements. Requirements Engineering is capable of bridging the gap between analysis and design of system as the output of such processes could serve as input to modelling design of systems. It starts with gathering user requirements, then performs analysis on the gathered requirement, and finally documents them in an appropriate format.

Following sections will cover the understanding and application of techniques that would help to formulate user requirement.

Requirement Elicitation

Requirement elicitation are concerns with identifying problems to be solved, what the user (or stakeholders) are trying to accomplish with the system, and how the system addresses the business need or generally the business model. The process begins by understanding and analysing the business problems. Business analysis often reviews that people within an organisation would have different needs (or opinions) and views concerned with the overall requirements of the system. They are stakeholders who either directly interact with or are indirectly affected by the system requirements. Hence, the focuses of requirement elicitation are to analyse the stakeholders' roles and how the operations that they performing – affects the system and hence must be specified in the requirements.

Stakeholder Analysis

Stakeholders are of primary importance to any project due to enormous project resource that has been invested to know exactly what the user wants. If stakeholders are approached earlier in the project, it is easier to communicate their requirement and work out their high priority concerns. The initial step to discover stakeholders requirements would be via Stakeholder Analysis. Stakeholder Analysis views a system as "a complex set of interacting elements which working together to satisfy needs or objectives". The idea is to discover how, when and where stakeholders are involved in the process. The different levels of stakeholders' involvement in the system can be viewed from a stakeholder analysis.

The analysis process starts by identifying the first group of people who form part of the system (in a hierarchical manner) namely the operators. They are referred to Admin and Users. They undertake specific tasks to achieve their operational goals and the system cannot operate without them. Following this, stakeholder analysis continues to discover the next group of stakeholders (in the hierarchy), usually beneficiary of the first group. The admin may be viewed as a direct functional beneficiary who practically can do everything in the system. In additions the user needs to fill in the form and submit it, by logging in with his User ID. Finally the admin has access to edit and manage all the data.

Identifying Stakeholder Operations

After the initial analysis of stakeholders, the next step is to understand the responsibilities of stakeholders. However, precisely identify the operations involved in each distinct roles is a prerequisite for detailed task analysis in design phases. The functions that support their operations usually are requirements of the system.

It is important to gain insight in to what kind of system should be implemented and the level of change that may affect organisation before determining which requirements are appropriate for a given system. Hence, the steps taken after gathering the initial requirement involve performing an analysis on information obtained. Some of the basic techniques that could be applied during this process as discussed by and are described in the following:

a. Classification and organisation of requirements, involves grouping related requirements and organises into logical clusters or modules. Using model of system architecture is a common way to discover possible modules (sub-system) and associate related requirements to them.

b. Prioritisation and negotiations, involves prioritisation requirements resolve conflicting requirements through negotiation with stakeholder. The concern is to achieve a set of agreed requirements that considered views of stakeholder involved.

c. Additionally, the analysis process may also involves business perspective such as business process automation, business process improvement and business process reengineering. Business process may not too be the focus of this report, but possible improvement and automation should be considered as system requirements.

Requirement Classification and Organisation

As specified above, organising requirements involves grouping requirement into related logical clusters to identify their relationship and dependency. Requirements may be vague at this stage because they are stated in the form of stakeholder's operational goals. Nevertheless, they could be translated into sets of required system functions. As for the complex software system, grouping related system functions often leads to a modularity view. Adopting modularity views in software architecture allows developers to have clear segregation of each subsystem concern and their relationships.

The result of grouping requirements is based on system functions. Each grouping is given a module naming. Additionally, it relates the operators (stakeholders) operation goals specified to system functions. The last concern in the table is the dependencies of each module, serving as important criteria when prioritising the requirements in the next section.

Prioritising Requirement

Prioritising requirement involves ranking requirements by weighting the characteristic of requirements in terms of user needs and dependencies. High priority requirements should be addressed first because other requirements often depend on them. These requirements would also fulfil basic operation goals of stakeholders. Prioritising requirement is an activity according to the principle of incremental planning. Requirements can be changed depending on the time available and their relative priority.

Based on module dependencies, they should have higher priority compare with other modules. However, each requirement maybe further broken down into sub-requirements for enhanced usability or improved workflow. They may not be essential to basic operations and thus would subject to lower priority. In brief, priority of requirements is concerned with balancing dependencies and user needs.

A method of prioritising requirements in the format of features lists, provides initial abstraction on the important requirements, followed by optional features. The list will require its priority to be further elaborated during requirement specification to achieve

optimum ranking. This is described in the functional and non-functional requirement sections.

Requirement Specification

Requirement specification aims to define requirements in clear and unambiguous language based on requirement identified during requirement elicitation and requirement analysis. The requirements are evolved over time and become more accurately reflect the needs of the stakeholders. The next steps are to document these requirements and treat them as ultimate references of domain knowledge and business rules. Documenting requirements can be challenging, and it is aligned to the fact that the software industry appeared to use the term "requirement" inconsistently. Requirement could be a high-level abstract description of system provided services, or comprehensive formal definition of system functional units. Given the vast difference of stakeholder's perspectives and interests about the best way of specifying requirements, a further investigation into approaches of documenting requirements is necessary.

Requirement documents which are also known as software requirement specifications (SRS), contain important statements describing the software product to be built. The level of details may vary depending on the type of developing system. Safety critical and complex systems often require detailed description of constraints or essential domain knowledge. On the other hand, requirements for commercial software are often changing and become out-of-date quickly. It appears that use case and story card from agile development methods are more flexible in capturing business requirements. Based on suggestion, the agile approaches in documenting requirement are:

1.Focus on software, not documentation. Create it only if it is essential to the work effort

2.Keep it simple. Create the most minimalist version of each artefact and use simple tools

3.Proceed iteratively. Start by identifying a high-level model and gather the details as the work proceeds

4.Collaboration could improve communication thus reduce need for documentation.

Following on from this, the next steps are looking at two major categories of requirements: functional requirements and non-functional requirements.

Functional Requirement

Functional requirements describe the features that the system has to provide. It often describes the expected features the user can utilise to perform their task. It may cover certain business processes and procedures that the software must follow and perform to

achieve user goals, sometimes termed business logic. Thus, Functional Requirements are often associated with desired behaviour characteristics of developing software to produce the expected result. The primary Functional Requirement of the system is to send forms to the database with information that are filled from the user.

Non-Functional Requirement

Non-functional requirements refer to characteristics and constraints with which the system must comply. It concerned with quality aspects of software system and good user experience such as performance, security, availability. Unlike FR, NFR does not usually related to functionality that yields operation results directly. However, NFR do affect the experience or result quality when the user using the system. Thus NFR may affect the overall architecture of a system rather than the individual components."

The supervisor further claimed that NFR might also generate FR given that the example of implementing Security features could introduce new system services. There are several concerns with respect to NFR, especially usability or usability analysis. Users should feel comfortably navigating the system using desktop or laptop devices. The user interface should provide clear indication of navigation path, and the menu, colour and layout should look consistent to the users. Security is also a major consideration when developing the system. Access control should be implemented to prevent destructive actions.

Context Model

During the initial phases of modelling requirements, developers must first understand the context where the system operates in a specific environment. The context model captures this perspective to allow decisions on project scope to be made by the stakeholders. The context model also shows system dependencies to its interacting environment. The external dependencies could be another automated system, manual processes, and functional peripheral or actors. They might produce data for system usage or consume the output of the system.

Context Diagram (CD) which depicts overview of system working environment is one of the techniques utilised to model the system context. It shows system interaction with external entities and is used to identify information and control flows among these interactions. The two fundamental components in CD are actor and message. Actor represents the external entities with which the system interacts. It refers to a particular user's role who uses the system to perform task or external systems that are required by system to provide functionalities. Message encapsulates information flow and controls as part of connection between system and actors. Each connection is labelled with information or particular functions that flow between actors and system. These

connections provide insight into possible events that the system must response if the message is a particular type of command. For instance, the system will send the information form to database when User submitted form information. A typical message will contain two essential properties: data content and arrival pattern. Data content depicts the information that the message carry while arrival pattern describes the nature of message occurrence and possible events that trigger its occurrence.

The list of actors is identical to the stakeholders described above as they are the primary operators of the system. The Context Diagram also included external actors that the system was interacting with, such as Database. The message flows show typical information used in the environment.

Example description of Form Information Message

Message : Form Information
Data Content Contains form associated with selected table and their authors.
Arrival Pattern Occurs asynchronously when user submit form based on their request.

Context Diagram allows a clear understanding of system dependencies and system scope. Besides, explicit labelling information flow among system and external actors provides initial concept of data structure and process sequence. These ideas will then contribute to design analysis and system architecture.

Use Case Model

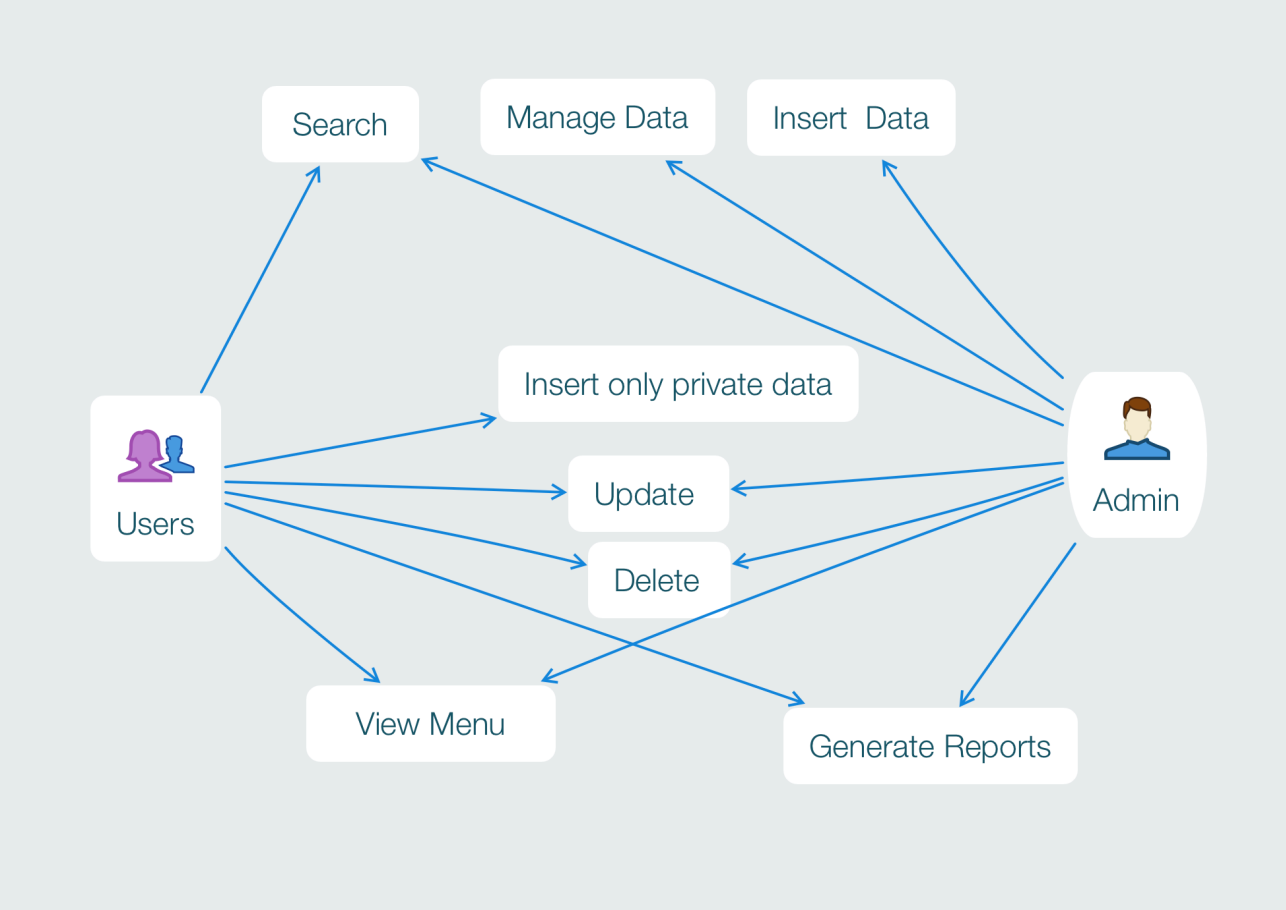
Use case modelling is one of the commonly applied modelling techniques in requirement modelling. Its primary use is to capture interactions between users with the system. Interactions that occur within a system could be user interactions such as input gesture, communication with external systems, or collaboration between components of the system. Knowing users' preferred ways to interact with the system also allows developers to capture precise requirements and build a more usable system.

Use cases are simple descriptions of system features from the point of view of users. Use cases also capture scenarios of what the user could perform with the system and the expected response from system. Nevertheless, a use case is often used to capture functional requirements of the system and generally are inappropriate for non-functional requirement. Use case modelling involves two major artefacts: use case diagram and use case description.

Use case diagram is a simple representation of what functions the system allows actors to perform. It provides a high-level view of the relationship between actors and functionalities. Each use case is represented as an oval shape and each actor is represented as stick figure. An actor could provide input and receive output from

associated use cases and these associations are depicted by line. The diagram shows all the actors that interact directly with the system features.

USE CASE DIAGRAM



Description Actors Scenario

User intended to create, update or delete private data.
The use case starts when user enters the menu after the user signs in. The system will present the menu. The user select a table of publication. And then selects what to do with the data inside that table (insert, update, view ,delete). Then the user submit the completed form and the system collects the updated data.Create and Update From will prompt user to input the details. Delete action will selected data from database. After selected action is completed, the user confirms to persist the result.

Use case diagram shows only a small amount of detail about the functionalities and their flows. Therefore, use case descriptions are required to provide in-depth information to understand what is involved. They could be simple textual descriptions, a tabular and structured description, or a sequence diagram. Abstracting the interaction details in the scenario and intentionally avoids premature decisions such as UI element design.

In summary, modelling interactions are fundamental steps to explore essential features required to achieve user objectives. This is particularly useful for grouping related features belonging to an actor role.

Based on the requirements discussed in previous sections, the project needs to formulate a project plan or sets of development activities derived from software engineering methodologies. As discussed earlier , the process agile method will be adopted as the software process model for the project. The important concern of adopting software process model is not strictly follow every principles and steps – but to use it as guiding principles. This section intended to cover some methods that could be used to improve traceability of project activities to requirements. It involves structuring tasks to be performed (i.e. software process model) and consideration of significant milestones.

Small Iteration or Releases

The development activities could be easily organised based on the system modules and features. Each modules depicts a major release of the software features that are then verified against a sub-set of the requirements. It is important to note there are various dependencies between these modules. Some lightweight tasks such as defining interfaces for other modules took precedence to enable development smoothness. At the end of each iteration, there are possible chances to re-evaluate requirements and adjust the plans, once each is: evaluated, tested, and validated against requirements. Each version of release prototype will be traceable to a FR or NFR. This ensures high priority works are focused on first.

2.4 System Architecture

As one of the major concerns in software engineering, the term software architecture could be understood through various definitions. Defined it as:

“The process of defining a structured solution that meets all of the technical and operational requirements, while optimising common quality attributes such as performance, security and manageability”.

In addition, defined software architecture by highlighting its important elements, listed as:

1) Highest level breakdown of a system into part

2) Decisions that are hard to change

3) There are multiple architecture in a system

4) What is architecturally significant is one that can change over a system’s lifetime.

These definitions mean that the software architecture phase represents the important design decision phase required to build a stable and scalable system. This is crucial for a successful system because it helps to address the quality and risk factors aligned to the project. Moreover, design concerns such as selection of algorithm(s), business logics and data structures often overlap with architecture decisions. Thus, the vision of architecture needs to be established to provide direction of development style.

This project is developing a website hence, the following sections will explore several software architecture styles for the website.

Client-Server Architecture

The most common architecture style for a distributed system such as website is the conventional client-server architecture. It divides the system into server components that offer services and client components that provide user interface to consume the services through connected networks. The server could serve request from multiple clients. This architecture style is also known as 2-tier architecture.

Clients are often represented by range of applications with GUI (Graphical User Interface) to capture user inputs and transform them into requests to the server. The server contains data storage to collect, modify and distribute data. Client could be either a thin client if application processing logic is located at server side or a fat client if it is embedded with client application. However, this architecture style lacks scalability because both the client and the server have limited resources. Reusing the web logic for different modules is also increasingly complex as systems expand due to their tight coupling between either data tier or presentation logic. In addition, it is difficult to maintain because it is hard to distribute new changes to system users.

N-tier and Layered Architectural Style

The N-tier (multi-tier) or most popular 3-tier architecture style addresses conventional 2-tier architecture issues by placing the application logic at an additional tier. Layered architecture shares the common goal with multi-tier architecture style, separation of functionalities into segments to improve scalability and maintainability. N-tier architecture is often used together with layered architecture style and they addressing following concerns:

i) Presentation logic (user interface)

ii) Business and website logic (domain process)

iii) Data accessing logic (database communication)

The presentation tier is at the top most level of the system and exposes visual representation to allow users to interact with the system. It gathers the user intention in the form of commands and inputs and forwards them to the business tier. The business tier is responsible for processing data between presentation and data tier. It contains important application logic performs calculations, evaluation and makes logical decisions. The result of the logic execution will either become responses to the presentation tier request or forwarding to data tier for persistent. At the bottom, the data tier handles create, update, read, and delete operations for the data sources. It contains various information to form appropriate queries for data retrieval. In an OO (Object Oriented) environment, it also in charge of handle mapping between information from data source to domain entity.

Based on the explanation the layered architecture style focusses on grouping related functionality into distinct layers and stacked vertically. Layered architecture model system structures in layers and each layer will have a collection of functions with similar concerns. Each layer can grow independently due to its loose coupling to upper and bottom layer.

In theory, multi-tiered and layered architecture styles have a number of similarities.

However, layered architecture has adopted OO design concepts into its core principles. Each layer promotes abstraction and encapsulation by providing just enough details to the dependant. It should also have high cohesion and clearly defined responsibility for maximised reuse. In contrast, multi-tiered architecture styles are concerned with the deployment of each segments into distributed environment. A 3-tier (layers) software architecture that is commonly adopted by web. As discussed in the previous section, N-tier architecture enables separation of concerns and simultaneously promotes scalability and loose coupling among interaction objects.

The Presentation Layer (PL) is responsible for mapping the information collected from domain entity to the UI element in HTML. In addition to this, it handles certain UI transitions on the client side to allow the application more responsive to user gestures. It is also important to ensure the presentation logic decouples from the business process logic. Separated presentation patterns such as Model-View-Controller (MVC) and Model-View-ViewModel (MVVM) will be utilised to manage communication between the

presentation components. These are discussed in details and with the framework used to implement them.

The Business Logic Layer (BLL) is a middleware layer that provides the services of business functionality for PL. It encapsulates different computational logic, such as calculating form total amount or workflow logic, such as updating a menu plan within each domain services. It also contains important business entities that serve as basic data structures to carry and process information. This layer employs an OO design and analysis approach to address the structural and behavioural aspects of the website.

The Data Access Layer (DAL) mainly handles communication between website with the data sources. Its responsibilities include establishing necessary connections to the data source and construct necessary queries to retrieve and persist data. The database context will take care of the life cycle of database connections while entity configuration will adjust database schema on when the entity structures change. The design of this layer is mainly concerned with data structure and data storage methods.

An overview of the physical architecture that will be supported by prototype software. The web, deployed into the web server, will provide centralised access to its services based on the local area network. In addition, a database server will hold the operation data of the system.

With both a vision of software and physical architecture established, the project has set a baseline for further design processes. The next steps will involve producing design models to address the design concepts of these architecture’s components.

Chapter 3 – Modelling and System Specification

3.1 System Modelling

System modelling is a process to construct abstracts representations of a system in several models, with each models encapsulate different views and analysis perspectives towards the system. In fact, it primarily focuses on creating design models. It works closely with requirement modelling by transforming requirement analysis result to design representation for building software. These models encapsulated requirement understanding such as specification of software operational characteristics. Software interface with other system elements and constraints that software must meet. Often, these design models could easily translated from functional and non-functional requirements and via versa. They are used throughout development process and they are commonly used for:

A.Facilitating discussions about existing or proposed systems
B.Documenting an existing system
C.Acting as a detailed system description, which could be used to generate system implementation.
Design models are often represented by different types of graphical notation with additional labels to describe their meaning. Depending on the development approach, different types of notation could be used to express a particular design. The most notable design notation used at present is UML, a unified modelling language that contains a robust notation for the modelling and development of object-oriented (OO) systems. Considering UML diagrams are de facto artefacts in the analysis and design process. However, this does not cover every representation or variant of design concepts. Additional diagrams such as Class responsibility-collaborator (CRC) Card and Entity Relation Diagram (ERD) will be adopted to express uncovered design concepts.
To follow the model with a purpose principle, models that are meaningful to the development process. While various types of models existed, modelling perspectives that are critical are:
a. Behaviour model, describes dynamic behaviour of a system and its response to events;
b. Structural model, describes organisation of system structure and how its information is been processed
c.Data model, describes information that will be persisted and their relationships.

Each model could employ several possible modelling techniques and artefacts to represent its perspectives. The project goal does not exhaustively produce every artefacts

of these models. It is rather to investigate how some modelling techniques could be applied to achieve the design objectives. The models also may not include every fine-grained level of details, yet these initial models will continue grow as developer refactors the design of the system.

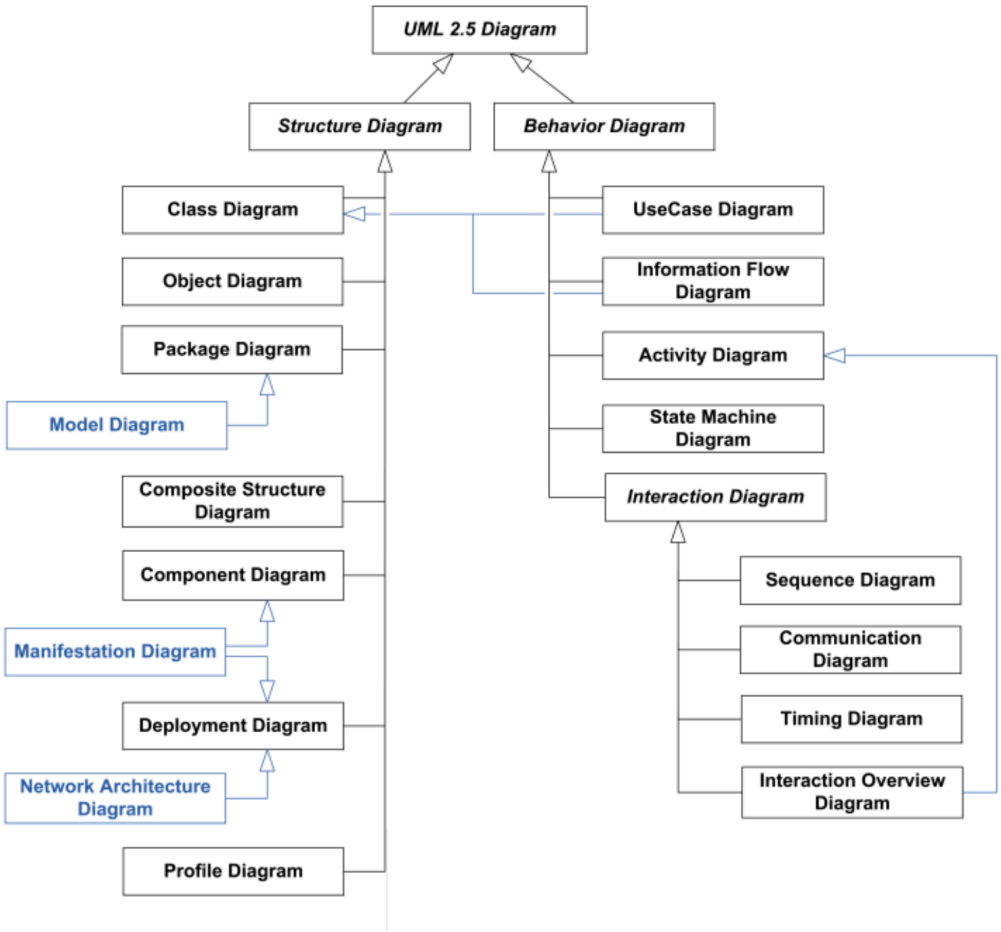
Structural Model

A complex system consists of several sub-components and possibly some external systems. Structural model displays the organisation of these components and their relationships. It embodies important consideration about entities that will operate within the system. In addition, structural model is to create a vocabulary that can be used by the analyst and the users. Things, ideas, or concepts discovered in the problem domain are represented as given object types in structural model, including relationships among such objects. In fact, this process is also known as domain modelling in OO development. Class responsibility-collaborator (CRC) cards and Class Diagram are the two focuses of the structural modelling techniques.

CRC card is a collection of standard index cards that have three major sections as specified below:

1) Class, represents a template used to create specific collection of similar objects;
2) Responsibility or operation
3) Collaborators, represents other interacting classes that are required for obtaining necessary information to fulfil responsibility.

3.2 UML State Machine Diagrams



Class name will present at the top, while responsibilities and collaborators will present at left and right panel respectively below it. CRC modelling is good option to explore structure and content of a particular class. Experimentally walking through use case with these cards could uncover design flaws such as missing properties, coupling and cohesion. The class knows some information and they are often translated into property, while other responsibilities will be become operations.

The established CRC model will eventually be translated into Class Diagram. The Class Diagram is a model that shows attributes and operations of system classes, as well as relationships among them. Class Diagram can have two levels of detail:

i) analysis class diagram

ii) design class diagram

The Analysis Class Diagram is used to understand the relationship between domain objects while the Design Class Diagram contains in-depth descriptions of object's attribute and operations. In fact, Analysis Class Diagram is also known as domain

modelling in OO development. It illustrated the possible elements that will be implemented as actual class.

Behaviour Model

Behaviour model aims to model the dynamic behaviour during the execution of the system. It can used to describe a use case at a specific time and event. Behaviour model is particular useful to model a possible business process. The business process can be expressed in a series of continuous interacting objects in the system. Suggested that there are two types of behaviour modelling.

The table shows a comparison of two types of approaches in behaviour modelling. It describes the purpose of modelling when they should be used and the tools that could help during the modelling phase. As a system to support business operations, expecting various explicit inputs from end users. For instance, when a User submits a form, the type of table is specified. Hence, data driven modelling is adopted because it fits the data processing nature. In addition, since the modelling process focuses on OO design, Sequence Diagram is used to model behaviour as opposed that is mostly used in a structured analysis design.

Data driven modelling

Shows the sequence of actions involved in processing input data and generating an associated output Useful when used to show entire sequence of actions that take place from an input being processed to the corresponding output Data-flow diagrams (DFD), and sequence diagrams.

Event Driven Model

Shows how a system responds to external and internal events.

System has a finite number of states and that events may cause a transition from one state to another. Sequence Diagram is one of the UML diagram to objects, including actors. It demonstrates a sequence of interaction activities during a system flow. The interaction activities could be a reflection of an explicit sequence of messages that have passed between objects [25]. All the objects will be arranged in a parallel line and a vertical dotted line indicates its active timeline. It is specified that not every detailed should be included in a Sequence Diagram unless it is used for code generation. The reason being it may lead to a lot of premature implementation design. Eventually, it could easily fall into entropy of big design upfront.

Furthermore a Sequence Diagram to capture behaviour to submit a new order. It consists of high-level view abstraction on the potential classes and messages exchanges required to be carried out to submit an order use case scenario. Messages flows will inspires operations that need to be implemented for the potential classes.

Data Model

Data model is concerned with exploring data-oriented structures [59]. It aims to define the structure of data objects, their relationships and related information that describe the objects and relationships. Data model is important because every application, at a certain point, will need to persist its data to certain type of storage. The structure of data, if not carefully designed, will affect data retrieval performance. The data model design is closely related with the decisions of data structures. Before this process can even begin, the type of data and model behaviour and interaction between storage strategy needs to be investigated and carefully selected to accompany the data processing need within the system.

The Context Diagram lets the designer (and/or the developer), realise a view of input and output data. The views are further developed into more concrete ideas regarding inner content of data, through structural models. This section develops the concepts related to the explicit data structure concern in this project.

The first issue, if data structure is concerned and related to this project, that needs to be addressed is: persistent or non-persistent. Once the decision of persistent or non-persistent has been made, the next issue is whether this data is going to be stored externally or internally to the application. This implies an external data store that will need to be selected, designed and implemented. In fact, it involves designing different types of data structure. If different types of data structures need to be designed, we need to think about different type of physical files that need to be processed. Hence, the next decision would be the type of files: whether they are simple or complex. If the file is going to be processed implicitly, it could be a text file, or comma-separated values (CSV) file. However, if the file required more processing and involves complex read-write operation, it will be more appropriate to select a formal mark-up file as the data store. This could be a data structure of relational model or hierarchy model (e.g. Extensible Mark-up Language).

3.3 Relational Database Management System (RDBMS)

RDBMS is data-processing software that employs relational model as its fundamental data structures. RDBMS describes the relational model as following in the relational model, all data is logically structured within relations (tables). Each relation has a name and is made up of named attributes (columns) of data. Each row contains one value per attribute.

It presents the data in tabular form identical to spreadsheet format. The standard language used for data manipulation in RDBMS is Structured Query Language (SQL). SQL consists of two major components: Data Definition Language (DDL) for defining the data structure and controlling access to data and Data Manipulation Language (DML) for retrieving and updating data. SQL can be very powerful depending on the usage. Some applications leverage its capability to transfer part of the system computation logic to RDBMS through writing stored procedures with SQL.

RDBMS has become the dominant data storage methods for most software systems today, particularly websites. There are many existing RDBMS solutions that all share similar sets of essential data processing functions varying slightly in the provided features. There are several mature commercial solutions such as Microsoft SQL Server [62] and Oracle Database. Conversely, there are also free open source solutions such as MySQL. These have been widely adopted by industry, particularly in small and medium business. The advantages of RDBMS are:

A. Support for controlling concurrency and transactional access;

B. Support for security management;

C. Minimal data redundancy

D. Simple user interface to manage data schema

E. Ensuring data integrity through constraints

F. Fast data retrieval through query optimisation

Extensive Mark-up Language (XML)

XML is a meta-language that allows designers to define their own customised tags in a document. As a type of semi-structured data, XML is designed to be self-descriptive,

readable by both humans and machines. It has a loose restriction of schema, thus allowing it to handle data structure that changes rapidly and unpredictably. This is especially true for information on the Web, where it requires certain degree of flexibility to accommodate ever- changing HTML design. The software industry today uses XML as the de facto standard for data communication. It has evolved to be the primary medium of data exchange with external systems and among businesses.

Many technologies have built upon XML by a predefined structured format for XML with XML Schemas. These schemas lead to standardisation of XML format when adopted widely by the industry. SOAP and RSS are some of examples of spin-off technologies based on XML. XML also has become popular thanks to a wide range of query languages available.

XML could be considered as the data storage if following advantages can be utilised:

A. Ability to deal with frequent and unpredictable schema changes;

B. Modelling hierarchy data structure effectively

C. Minimum data conversion if data is used directly from source; and

D. Support for multiple platforms.

Storage Method Chosen

The storage method chosen for both persistent files (Application Setting and Business Entity) is the RDBMS. The main rationale was the Business Entity, primary data structure, required extensive cross-referencing. For instance, an order needs to know which recipes been selected and the recipes need to know what are the materials that need to be consumed. Putting this data into hierarchical models will result into redundant data everywhere. RDBMS is also has strong security measures (access control) and they are important for Applications/Websites Setting, which contains critical data that would affect entire system behaviours.

In addition to that, another key factor that leads to this decision is the existence of Object Relational Mapping (ORM) solutions. ORM solution mainly help in converting the relational model into interconnected object graph, coined as the “Object-relational Impedance Mismatch” problem. This significantly reduces the complexity of retrieval

relational data into OO environment because complex SQL queries could be simplified into normal OO method calls.

Conversely, using XML as data storage could be relatively complex and verbose. The processes to retrieve the data in documents, map them to object and primitive data type in programming language, is difficult. Because the data type constraint is not enforced within XML, the designer will need to handle various kinds of data processing issues such as null value and format mismatch.

Entity Relationship Diagram

Entity Relationship Diagram (ERD) is widely used data model to represent relational data model of database. It is a picture which shows the information that is created, stored, and used by a business system. There are three major concepts in ERD. First, each data object in ERD is a named entity, often mapped to a table in RDBMS. Second, information with an entity is represented by a set of attribute, which capture the data segment (table title) of a data object (author). Finally, association among entities, also known as relationships, depicts high level business rules of a system. This model is reflecting actual implementation in the database for data persistence.

Chapter 4 – Data Representation

4.1 Description of Database

Database is a collection of logically related records (or data). It may consist of enormous number of data. Hence it is really important to manage and organise data within the database. Database Management Systems (DBMSs) are software applications that use to interact with database and the user by giving capability of managing data. It allows tagging, retrieving and manipulating data efficiently and quickly while ensuring the security and unauthorised access. SQL Server is one of the well-known DBMS used today. Web-based System uses a database for storing and managing records of loans and reservations, equipment details and user details. It uses basic database interactions such as insert, select, delete and update in different scenarios (for example when registering equipment or user, displaying equipment details, modifying and deleting equipment or user details, inserting loans and reservations).

4.2 Apache Server

The Apache HTTP Server, informally called Apache, is the world's most popular web server software that in 2009 it became the first web server software to serve more than 100 million websites. The Apache development began in early 1995 and originally based on the NCSA HTTPd server. Apache is developed and maintained by an open community of developers under the patronage of the Apache Software Foundation. Mostly used on a Unix-like system, the software is also available for a vast variety of operating systems, including Microsoft Windows, Open VMS, eComStation, NetWare and TPF.

Apache is open source software, as on November 2015, it was estimated to serve 50% of all active websites and 37% of the top servers across all domains.

4.3 MySQL System

SQL stands for Structured Query Language. MySQL is an open source Relational Database Management System (RDBMS). It is a popular database for use in web applications, and is a central part of the greatly used LAMP (Linux, Apache, MySQL, Perl/PHP/Python) open-source web application software stack.

MySQL is used by many applications like, WordPress, Joomla, TYPO3, Drupal, MyBB, phpBB, MODX and other software. Numerous large scale websites including Google, YouTube, Facebook, Twitter, and Flickr are also using MySQL.
On all platforms excluding Windows, MySQL sends with no GUI (Graphical User Interface) to administer MySQL databases or managing the data held within the databases. Users may install MySQL Workbench by downloading separately or simply may use the command line tools. Numbers of third party GUI tools are also available.

Swedish company has created MySQL which is written in C and C++. The first version of MySQL revealed on 23 may 1995. It has various versions. The general accessibility of MySQL 5.7 was broadcast in Oct 2015, and the version which is used in my project is 5.6.17.

4.4 List of Database Tables

Database Tables

JOURNALS TABLE

Author(s)
Paper title
Journal name
Volume
Number
Pages
Date

BOOKS TABLE

Author(s)
Title
Publisher
Place (Optional)
Pages (Optional)
Year

CONFERENCE TABLE

Author(s)
Title
Conference
Pages
Date

BOOKS CHAPTERS TABLE

Author(s)
Title
Pages (Optional)
Editors
Publisher
Place (Optional)
Year

OTHER TABLE

Author(s)
Title
Appears in: (volume ,number, pages)
Pages
Date

PATENTS TABLE

Author(s)
Title
U.S. Patent
Date Issued

Chapter 5 – Design and User Interface

5.1 Design Principles

System design involves designing a system based on user requirements, and testing the feasibility of the project. In general, the steps for a prototype system design include: database design and populating, system architecture design, user interface design and functionality development

Analysis model

Often a design element corresponds to many requirements, therefore, we must know how the design model satisfies all the requirements represented by the analysis model.

Programming paradigm

A programming paradigm describes the structure of the software system. Depending on the nature and type of application, different programming paradigms such as procedure oriented, object-oriented, and prototyping paradigms can be used. The paradigm should be chosen keeping constraints in mind such as time, availability of resources and nature of user's requirements.

Uniform and integrated

Software design is considered uniform and integrated, if the interfaces are properly defined among the design components. For this, rules, format, and styles are established before the design team starts designing the software

Flexible

Design should be flexible enough to adapt changes easily. To achieve the flexibility, the basic design concepts such as abstraction, refinement, and modularity should be applied effectively.

Minimal conceptual (semantic) errors

The design team must ensure that major conceptual errors of design such as ambiguousness and inconsistency are addressed in advance before dealing with the syntactical errors present in the design model

Degrade gently

Software should be designed to handle unusual changes and circumstances, and if the need arises for termination, it must do so in a proper manner so that functionality of the software is not affected

Correspondence between the software and real-world problem

The software design should be structured in such away that it always relates with the real-world problem

Software reuse

Software engineers believe on the phrase: 'do not reinvent the wheel'. Therefore, software components should be designed in such a way that they can be effectively reused to increase the productivity.

Designing for testability

A common practice that has been followed is to keep the testing phase separate from the design and implementation phases. That is, first the software is developed (designed and implemented) and then handed over to the testers who subsequently determine whether the software is fit for distribution and subsequent use by the customer. However, it has become apparent that the process of separating testing is seriously flawed, as if any type of design or implementation errors are found after implementation, then the entire or a substantial part of the software requires to be redone. Thus, the test engineers should be involved from the initial stages. For example, they should be involved with analysts to prepare tests for determining whether the user requirements are being met.

Prototyping

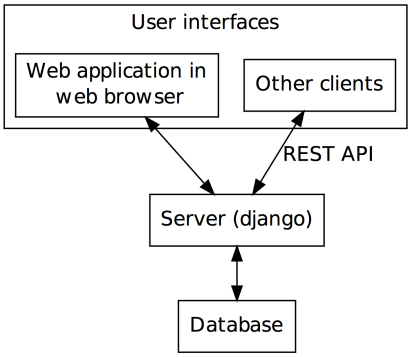
Prototyping should be used when the requirements are not completely defined in the beginning. The user interacts with the developer to expand and refine the requirements as the development proceeds. Using prototyping, a quick 'mock-up' of the system can be developed. This mock-up can be used as a effective means to give the users a feel of what the system will look like and demonstrate functions that will be included in the developed system. Prototyping also helps in reducing risks of designing software that is not in accordance with the customer's requirements.

Note that design principles are often constrained by the existing hardware configuration, the implementation language, the existing file and data structures, and the existing organisational practices. Also, the evolution of each software design should be meticulously designed for future evaluations, references and maintenance.

5.2 System Design - NOT FINISHED

System Design

5.3 Preview of User Interface



UI Design remains one of the main topics of the project. The UI concerns of this project are mainly designing UI that comfortably map user task to the interaction action, and more mobile friendly website design. The following section will introduce a design approaches called Task-based UI design. In the next diagram it shows the connection between the web browser the server an the database of the system.

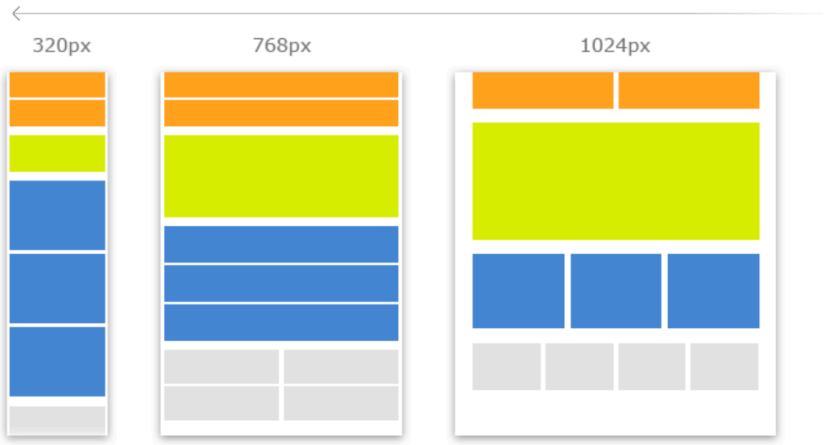
Introducing Task-Based UI Design

UI design has been increasing challenging for developers due to the rise of complexity and need for consistency. One possible factor that has contributed complexity is the users' mental model mismatch with the conceptual design of UI designers. Users tend to focus on the task in hand rather than struggling to learn the features and procedures necessary to activate services. Besides, it is also challenging to maintain consistency when more features are added to the UI. Users potentially distracted from finding desired command if many possible selections are provided. Thus, new kind of user interactions that are self-explanatory and has a minimum learning curve is required.
Most of the current UI design focuses on the viewpoint of domain objects, which hold underlying data that will be presented in the UI. Users have to learn the navigational system, the naming of interaction objects, and follow a series of navigation on the domain-oriented menu to achieve particular goal. The original intention of users has switched from simply submitting a form to interacting with several domain objects. There are additional layers of transformation from user intention to following the structure of system objects. A Task-based UI employs a different approach by presenting available commands that meet user goals. Using the previous example, taking submit button function will present a more task-oriented command such as "take form information" and arrange the interface to allow to send the data in the database.

Task-Based UI design

Task-based UIs provide better performance in achieving user goals. It is also less cognitively demanding as it allows direct mapping of interface commands to user tasks. Task analysis is an essential technique to model user tasks and system behaviour in Task-based UI design. It helps developers to understand possible user actions and subsequently transfer them into UI designs.

The next approach to this problem is Responsive Web Design. Responsive Web Design has gained momentum since the introduction of HTML5. With the help of the Cascading Style Sheet (CSS), a HTML formatting framework, it is possible to design a web UI that will adapt to various screen resolution sizes. It scales down or hides certain UI contents to improve presentation on different desktops. While this approach reduces significant efforts of maintaining redundant sites, it also increases the burden of web browser. As specified less visible content does not imply less content has been downloaded.



Recent website developments lean towards the responsive design approach. This mainly because it has received increasing support from JavaScript web frameworks such as Bootstrap, jQuery UI and Less Framework. These frameworks adopt a fluent grid concept to layout contents on different screen resolutions. In fact, this project favours this approach because it requires smaller overheads to maintain separate sites and possible to achieve maximum reusability.

Menu Design

After taking account the previous design approach we can clearly see the menu of the website. Which is a scalable design for different screen resolutions.

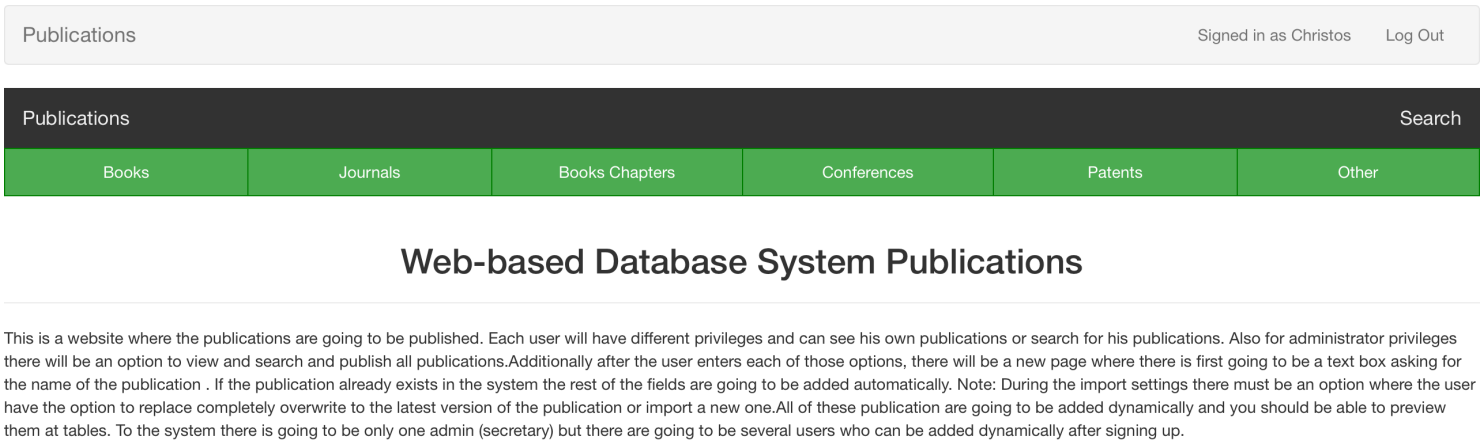
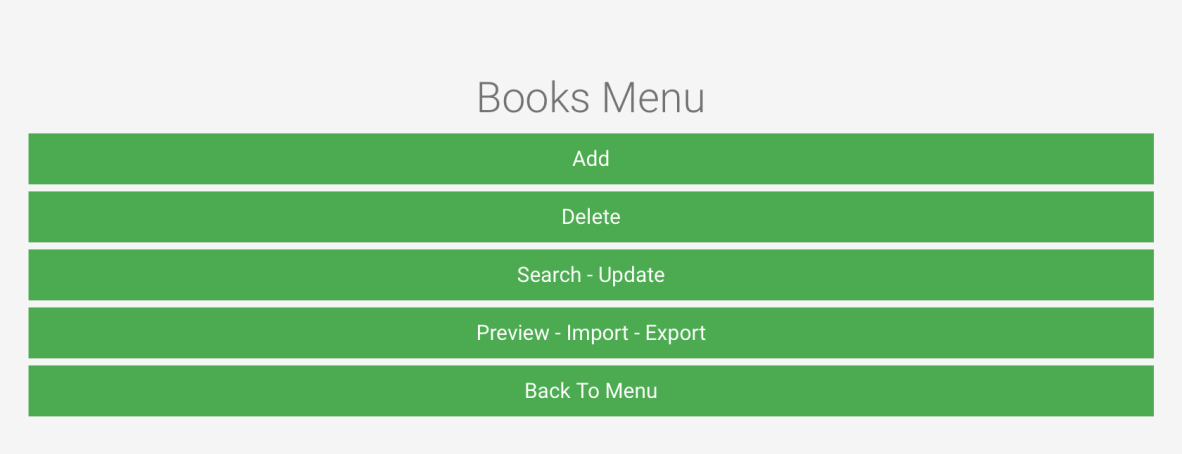
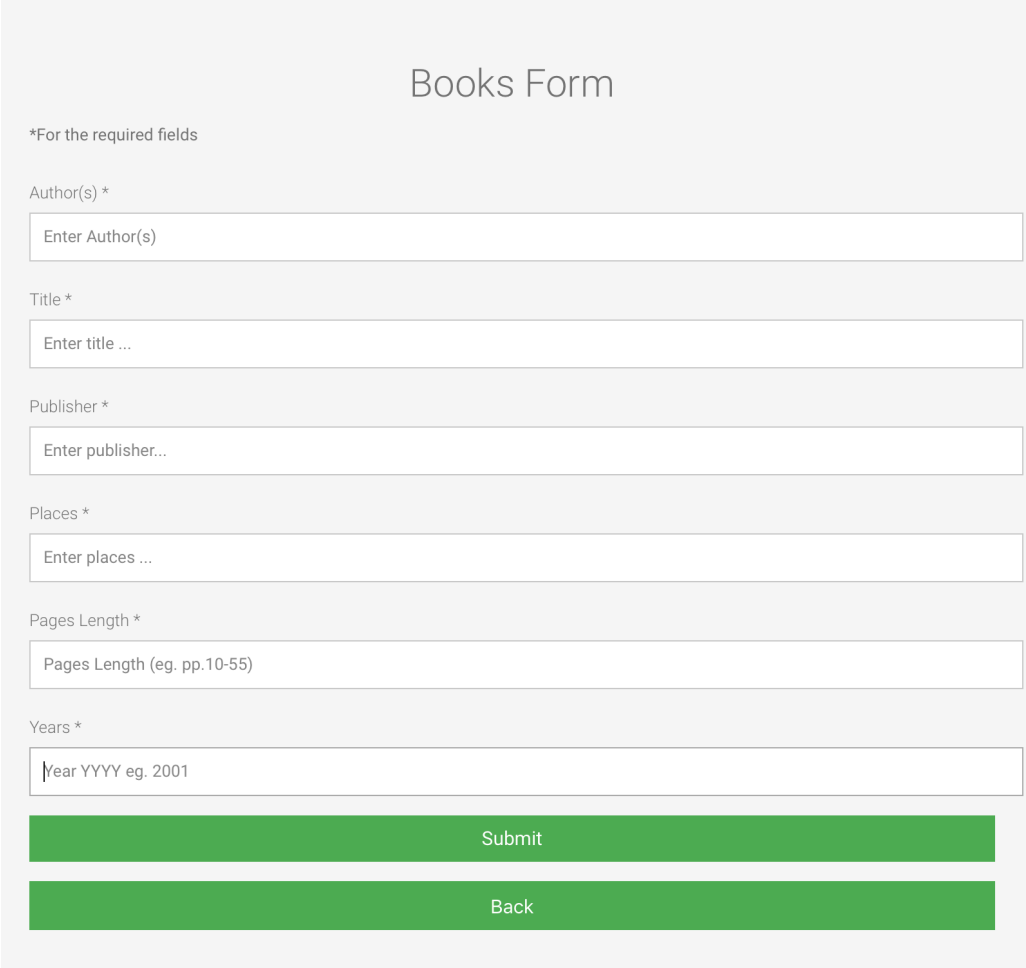


Table menu for Books



Form Design for Books



Chapter 6 – Programming Overview

6.1 PHP

It stands for PHP: Hypertext Preprocessor but, originally stood for Personal Home Page. Is a server side scripting language that designed for web development, as well as used for general purpose language. It was created in 1994 by Rasmus Lerdorf, in the present time the reference execution of PHP is produced by the PHP group.

In January 2013, PHP was installed on more than 240 million websites, and 2.1 million web servers. The PHP code can be combined with several web frameworks and templating engines or simply it can be mixed with HTML code.

The PHP code is generally processed by a PHP interpreter, which is commonly executed as native module of web server or a Common Gateway Interface (CGI) executable. After interpretation and execution of the PHP code, the results will be sent by web server to its client.

There are many versions of the PHP, and the version, I have used for my system is the **PHP version 5.5.** But a wise question would be the **reasons to choose PHP**.

Well, I chose PHP because it has interactive features and allows you to interact with your visitor in ways HTML alone could not. So for example simple things like email forms or like shopping carts that save your previous orders and suggest other products. It supports social things like interactive forums and private messaging systems.

Additionally PHP is easy to learn and a lot easier to get started. By learning just a few simple functions, I was able to do a lot of things with the website.

Furthermore works great with html, because I have already built some websites in the past and I am familiar with HTML, making the step to PHP easier. PHP and HTML are interchangeable within the page. PHP might add some new features to my website, but the basic appearance is still created with HTML. PHP is highly learnable. PHP is used for web oriented scenarios that need to be developed quickly, and need to perform really fast. In addition PHP sports an enormous community. Majorities are new to PHP but once you learn the basic, you can reach dedicated professional like Phil Sturgeon, Josh Lockhart. There are excellent resources like Site Point's forums, Stack Overflow and PTRW, problems encountered in PHP are easily solved, and progress is usually both rapid and highly educational. Significantly important is the fact that the number of open source PHP projects as well as books and courses someone can learn are plenty and with some patience and guidance you could become proficient in this language in a relatively short

period of time. PHP significantly has matured since a period of time people tried to bashed it, and is now inheriting more and more modern features from other languages. Prejudice is prevalent and an individual should learn to recognise it and appreciate it. “It’s not the tool, it’s how you use it”. PHP is now used on more than 80% of the world web servers. It has been helped in no small part by Word press.

Open Source

PHP is open source, primarily aimed at web development and applicable to similar projects. The code of PHP can be placed in any file which is interpreted by the PHP engine, typically once with a .php extension. You just enter the URL which maps to that file in your browser and you’re done. Admittedly, the code will only run via a web server with PHP installed. PHP has a built-in server. Although is even easier upload your file to almost any web host. PHP is conceptually simpler than any other language and is currently winning. Knowing a few PHP statements can write something useful. It has more software dependencies, but PHP concepts are less daunting to new developers.

Deep code base

The Web is filled with PHP code. The most popular platforms for building websites (WordPress, Drupal, Joomla) are written in PHP. Not only are the platforms open source, but so are most of their plug-ins. There's PHP code everywhere, and it's waiting for you to download it, modify it, and use it for your needs.

Simplicity

There's not much to PHP: a few variables and basic functions for juggling strings and numbers. It's a thin layer that doesn't do much except move the data from port 80 to the database and back. That's what it's supposed to do. A modern database is a magical tool, and it makes sense to leave the heavy lifting to it. PHP is the right amount of complexity for a job that's not supposed to be complex.

No client app needed

All of the talk about using the same language in the browser and on the server is nice, but what if you don't need to use any language on the browser? What if you ship the data in HTML form? The browser pops it up, and there are no headaches or glitches caused by misfiring JavaScript threads that try to create a page on the browser from two dozen Web service calls. Pure HTML works more often than anything else, and PHP is optimized to create that. Why bother with JavaScript on the browser? Build up everything on the server and avoid overloading that little browser on the little phone.

SQL

PHP was built to co-exist with MySQL and its many variants, like MariaDB. If MySQL isn't exactly right, there are other great SQL databases from Oracle and Microsoft. Your code can switch with a few changes to your queries. The vast SQL world doesn't end at its borders. Some of the most stable, well-developed code will interface with an SQL database, meaning all that power can also be easily integrated into a PHP project. It may not be one perfect, happy family, but it's a big one.

Speed of coding

For most developers, writing PHP for Web apps feels faster: no compilers, no deployment, no JAR files or preprocessors -- just your favourite editor and some PHP files in a directory. Your mileage will vary, but when it comes to banging a project together quickly, PHP is a good tool to use. PHP is a well tested technology.

Easier to mix Code

PHP makes it easy for the developers to mix code with content. One can simply open PHP tags and write codes without any need of templates or other files. It is designed to give the programming power at the fingertips of a developer.

Deep History

PHP code has a deep history in web development. All the major CMS platforms, like, WordPress, Joomla and Drupal, including their plugins are all written in PHP, making it easy to use and modify it according to one's needs.

Simple Language

PHP is a simple and thin layer language with less variables and certain elementary functions. It's a perfect option for a job that doesn't require much complexity.

Supports MySQL

PHP supports MySQL and its other family members like MariaDB. By interfacing most stable codes with SQL database, PHP empowers its projects with extreme flexibility.When it comes to delivering a project quickly, PHP is usually the developer's first choice. Without using any JAR files and compilers, it enables a developer to create a web application with simply an editor and PHP files.

Widely Used

I have learned PHP because it is the most widely used, and was designed for back end server programming.

Why not Node.js

Node.js is far more recent, less commonly used, is bases on JavaScript which is quirky and unusual in its design. It is not really designed to be good for server side programming, but is a compromise for programmers who are more familiar with client side methodology, who wanted to keep the same language features when doing server side. And I think that is a bad ideal, because I don't think JavaScript is a good language at all, for client or server side.Although, Node is providing the best possible solution, but there are always two sides of the mirror. Wisely think when making a Node application because Node.js is not suitable for processor intensive tasks and any CPU-intensive code makes it really non-scalable.Majority of the web's hosting providers are able to provide PHP hosting.With PHP, you can simply install WAMP, LAMP or MAMP and off you go. For deploying code into a web host, you just drop your files there and you're done.As PHP code runs each in its own process and the web server sits in front and directs this, if one of the requests causes an error it will only affect that specific request.In PHP each process is only alive for the duration of the request. This means you don't need to worry as much about resource allocation and memory.PHP's standard library is much bigger than Node's.

6.2 HTML

HTML stands for Hypertext Markup Language and CSS stands for Cascading Style Sheets are the crucial technologies for creating web pages. HTML supplies the structure of the page, and CSS the layout, for diversity of devices. Together with scripting and graphics, HTML and CSS are the fundamental of building Web Applications and Web pages.

HTML provides designers and developers the following facilities,

a. To design forms for directing transactions with remote services, for use in making forms, searching for information, and others

b. Retrieving online information through hypertext links.

c. To include video and sound clips, spread sheets, and other applications straight in their documents

d. Designer can publish online documents with text, headings, tables, photos and others.

e. CSS describes the Web pages presentation, involving layout, colours, and fonts. It enables the designer to adjust the presentation to various types of devices, like a small screens, large screens, or printers.

f. CSS is separate from HTML, and their separation makes it easy to preserve and maintain sites, share style sheets across pages, and accommodate pages to various environments

6.3 PHPMyAdmin

It is an open source tool and also, it is free written in PHP, XHTML, CSS, and JavaScript planned to manage the administration of MySQL by using of a web. It is able to perform various missions like creating, modifying databases, tables, fields, executing SQL statements or managing and supervise users.

PhpMyAdmin is being translated into 72 languages in order to make the usage easy to a wide domain of people and it supports both LTR and RTL languages.

Following is some features of the phpMyAdmin:
A. It is web interface
B. It administrates multiple severs
C. It is able to create PDF graphics of the database layout

D. Importing data from SQL and CSV

E.Export data to different formats such as SQL, PDF, CSV, XML and others

F. It works with various Operating Systems

6.4 Javascript

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Client-side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.
The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.
JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are:
1.Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
2.Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
3.Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
4.Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

6.5 Solution to Selected Problems - NOT FINISHED

Chapter 7 - User Manual

7.1 Website Overview
7.2 High Level View
7.3 Compatibility and Accessibility
7.4 Implementation
7.5 Code
Conclusion
Bibliography
Appendix

Figure for Research