Chapter 4

4. System Design

4.1. Introduction

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This system design section of this CSE Department Academic Resource Management System (CRMS) contains a detailed description of the objects defined in the previous chapter; the software requirement section. It will attempt to define methods, properties, accessories and to a certain extent, provide algorithms or ways of approaching the coding or programming process.

4.1.1. Purpose of the System Design

This software design section describes the architecture and system design of CRMS showing what methods and directions to be followed in order to develop the system. It expresses the essential building blocks that are necessary to build the system. It also provides the basic implementation aspects of any software. As this system is primarily an academic resource management system; it should be equipped with some sort of structure and organization throughout the system, such that the CRUD (Create Read Update Delete) rule as well as MVC (Model View Controller) layout must be maintained. As this standardized methods are implemented, the system would be open to highly qualified functionality, refactoring, update and upgrades.

4.1.2. Design Goals

The design goals describe the major qualities that a system needs to achieve and address through the design process. Generally, there are a lot of design goals to be achieved, yet the following are some of the design goals that are attempted to be attained for this particular system; namely CSE department's academic resource management system.

a. Reliability
   As the system is a resource management system; those resources should be reliable as well as the system. The system should provide reliable service to the users meaning providing the proper and authentic material in a very reliable, convenient, effective and efficient manner. It should be a well trusted source and portal of resource in the department's community.

b. Maintainability

The system should always be open to monitors and changes. If a certain feature in the system needs to be modified, that modification should address only that specific feature without affecting the overall functionality of the system which can be achieved by using the full stack development method as well as a framework approach.

The System should also be capable to add or enhance a certain functionality, it should be achieved by refactoring the existing system and bring about that change through minimal effort and inconvenience.

Adaptability is another aspect of maintainability which addresses issues like new releases through other platforms. The system should be simple enough to be adapted or outfitted into a new platform; this case is also being practiced as the CRMS web system is being adapted into an android mobile application in the near future, which also attains the portability goal as well since the app will be more reaching and easy for access.

c. Scalability

Scalability is also another major goal that needs to be attained through the design process, since the system can be scaled into more departments, schools and the entire university. So it should follow a certain set of parameters in order to assert this issue by using a more general, objective and broad classes and categories to maintain a scalable standard.

d. Robustness

As any given system, flaws and erroneous activities might occur and this must be addressed by redirecting to an error page, provide troubleshooting option and providing contact page, through all this the system should not crash or fail, it should handle such errors as it is supposed to.

e. Performance

The system should operate with in optimum response time and memory space. The system should respond to requests in a maximum duration of 10 to 20 seconds. It should also meet a certain memory and storage standard since it provides resources; that's why as a primary storage a state of the art cloud storage (Google Drive / One Drive) is used since it operates on a higher performance rate. There is also a secondary storage site that is the Web server's storage.

f. Low Cost

Since the system is developed by students with no proper budget in place, it should be developed and deployed within reasonable and low cost. This can be achieved by using free development methods like open source web & framework development methods and free domain name and also free web hosting services.

g. Understandability

The system should be well understood by the users and satisfy their need as much as possible. So in order to achieve that it must be presented with proper language, proper grammar, unambiguous format and user-friendly interface.

h. Rapid Development

As the system follows the rapid development model (RAD), it should be developed at an earliest convenience that is less than three months, so any tool that is deemed fit to achieve this would be utilized such as scaffolding tools, basic frameworks and simpler layouts.

i. Flexibility
   The system should bring a well thought out plan to implement a certain task; for instance provide a search option, perform single and bulk operations, and provide a navigation and directive.
j. Well Documented
   The system should be well documented especially through the design process since it provides the very fundamental information about the task commenced to develop the system. So it should have a proper documentation as well as a final product documentation that can serve as a user manual.

Design Trade-offs

Efficiency vs Portability: As the system tries a more efficient state, it would lose its portable features in the process. In order to provide an efficient experience, CRMS would lose its portable features, since it would consist a more constrained and predetermined path of operation.

Rapid Development vs Functionality: As the system is developed in a very short period of time, it would miss some functionalities and also discard some features on the way to meet the deadline.

4.1.3 Definitions, Acronyms and Abbreviations

Cloud Service Provider (CSP): A business entity that offers computing, storage, or software services powered by VMware cloud platforms to consumers through a private or public network.

Consumer (or customer or end user): Someone who consumes the services offered by the client or directly by the system.

Service: A means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks.

Cloud Management Platform (CMP): A suite of integrated products that provide management for public, hybrid, and private cloud environments (includes self-service interfaces, provisioning systems, metering and billing, and workload optimization).

ACS Auto-Configuration Server

SQL Structures Query Language

API Application Programmer Interface

ASF Apache Software Foundation

GUI Graphical User Interface

4.1.4 References

At the end of the documentation.

4.1.5 Overview

The following sections of this system design chapter would consist the software architecture- high level software architecture representation, the system design process, the subsystem decomposition, API layouts, hardware and software mapping, database setup, access control, boundary and constraint conditions as well as subsystem services.

4.2. Current Software Architecture

Software architecture refers to the high level structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the design teams.

This CSE Department Academic Resource Management System is accessed by the users through web browsers connected to the internet and the website provide all the necessary menus, navigations and materials from the webserver as well as the cloud storage provider (CSP).The system's architecture could be explained as follows:

- ✓ The website would provide a user interface or web page for ordinary users, instructors and administrators. This front end web system provides information and services to the users as well as a content management system for instructors and administrators.
- ✓ The webserver would contain the web site data, domain, subdomains, configurations, a storage space as well as a database that hold the information of all activities and services.
- ✓ The cloud storage would provide a storage space for the resources that are managed and organized by the system (CRMS) with capability to generate links to acquire the resources from the website.

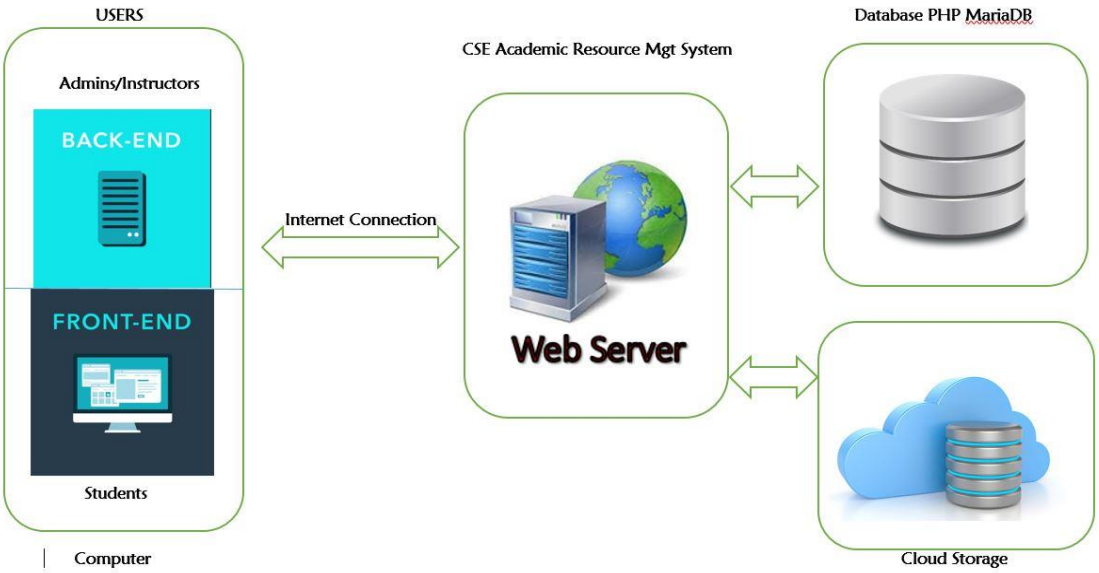

Figure 4.1System Architecture

4.3. Proposed Software Architecture

4.3.1 Overview

The system architecture can further more be represented by the following diagrams. It shows the entire interaction among the users (Students, Instructors, and Administrators) and the web system and also between the cloud storage and databases.

Client Server Communication

The client while trying to get the services of the system sends http requests to the website server which is analyzed through a parse PHP decoder. It also uses a composer JSON object to perform operations at the backend resulting changes at the database smoothly provided by the ASF with

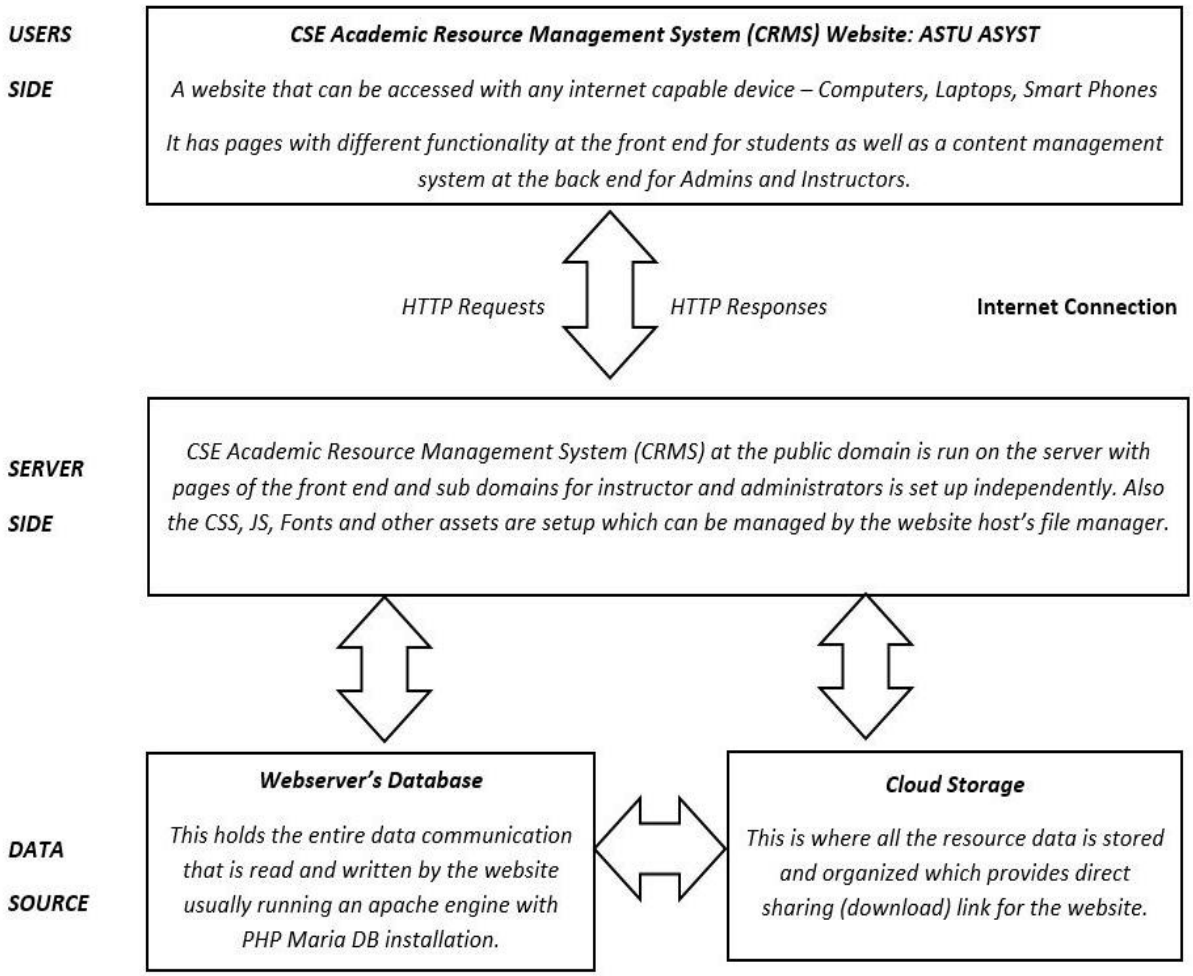

Figure 4.2 System Tier Architecture

Auto Configuration Server (ACS).

```php
$uri = urldecode(
    parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)
);

// This file allows us to emulate Apache's "mod_rewrite" functionality from the
// built-in PHP web server. This provides a convenient way to test a Laravel
// application without having installed a "real" web server software here.
if ($uri !== '/' && file_exists(__DIR__.'/public'.$uri)) {
    return false;
}

require_once __DIR__.'/public/index.php';
```

Then that request is processed through a pre-configured environmnet protocol, verified and suddent executed accordingly.

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=SomeRandomString
APP_URL=http://astuasyst.tk

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=288412_astuasyst
DB_USERNAME=astuasyst
DB_PASSWORD=********************

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Figure 4.3 Server Environment Configuration

System Process

The following diagram represents of a system process of how a front end user; Student can access the system and get services.

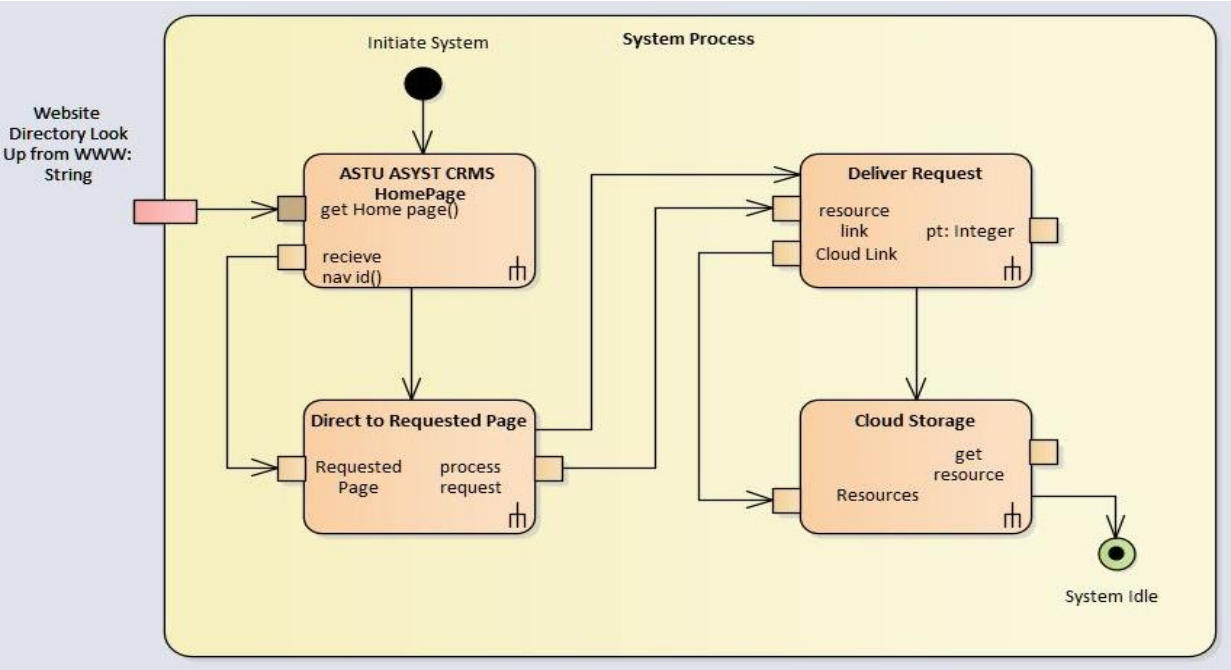

Figure 4.4 System Process (Front End)

The following diagram represents of a system process of how a back end / content management system user; Admins or Instructors operate and make changes to the system.

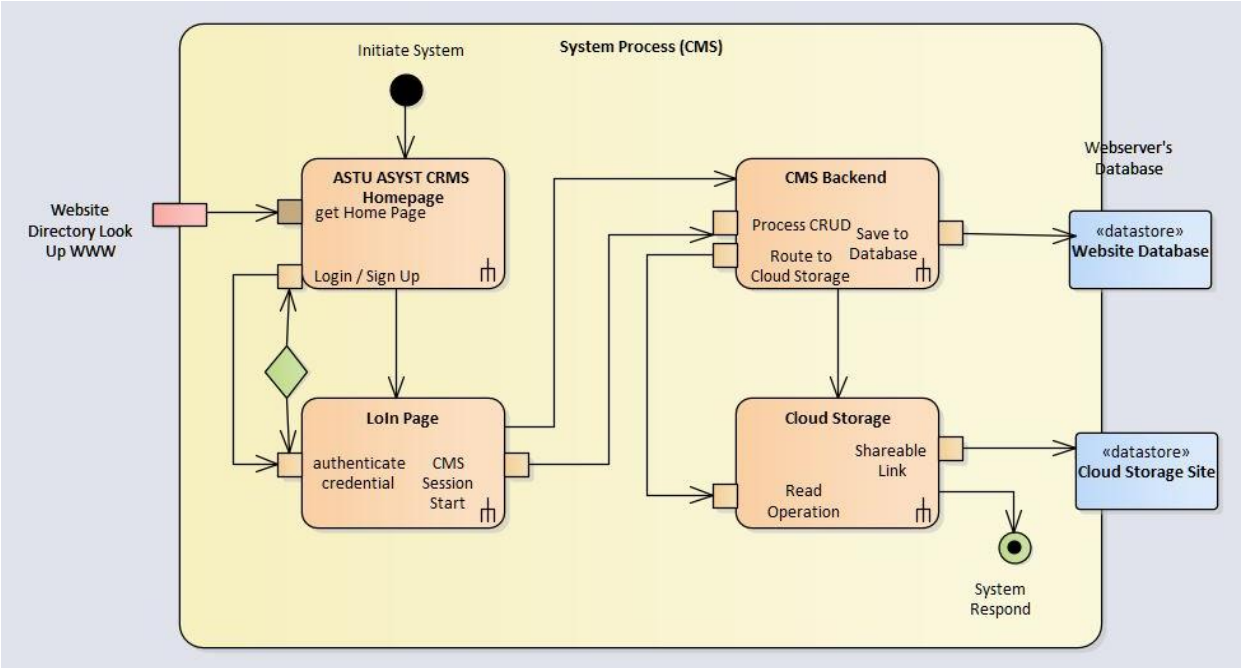

Figure 4.5 System Process (Back End CMS)

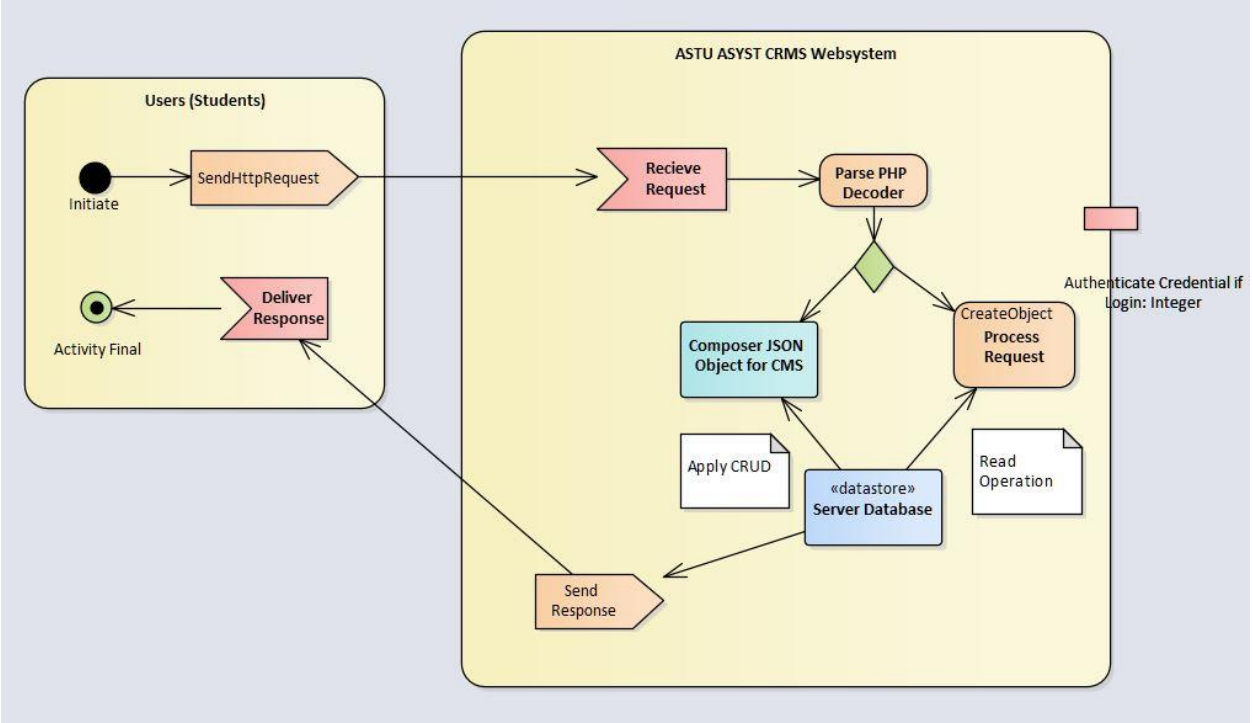

Figure 4.6 Overall System Process

4.3.2 Subsystem Decomposition

Subsystems are collection of classes, associations, operations, events and constraints that are closely interrelated with each other. The objects and classes from the object model are the "seeds" for the subsystems. In UML subsystems are modeled as packages. This system comprises of twelve (12) major sub system.

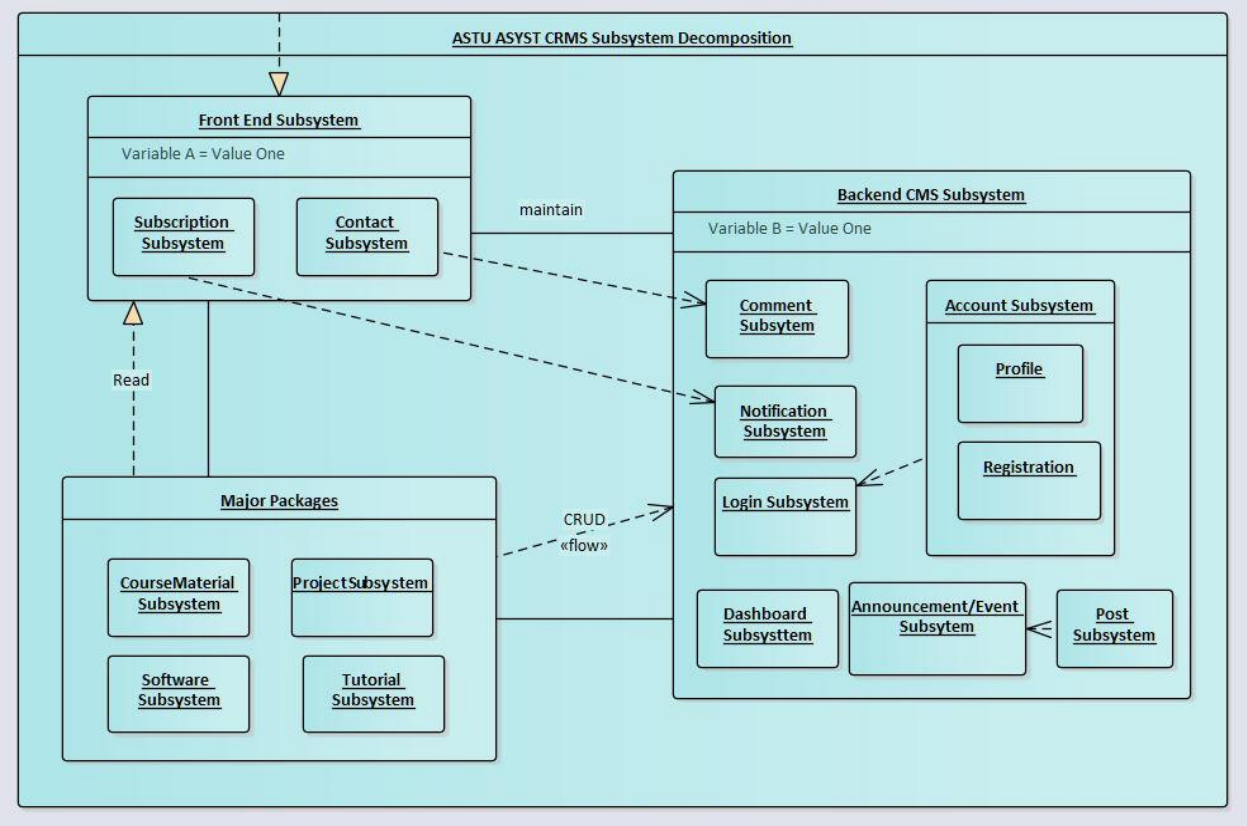

Figure 4.7 Subsystem Decomposition (Environment-Component Format)

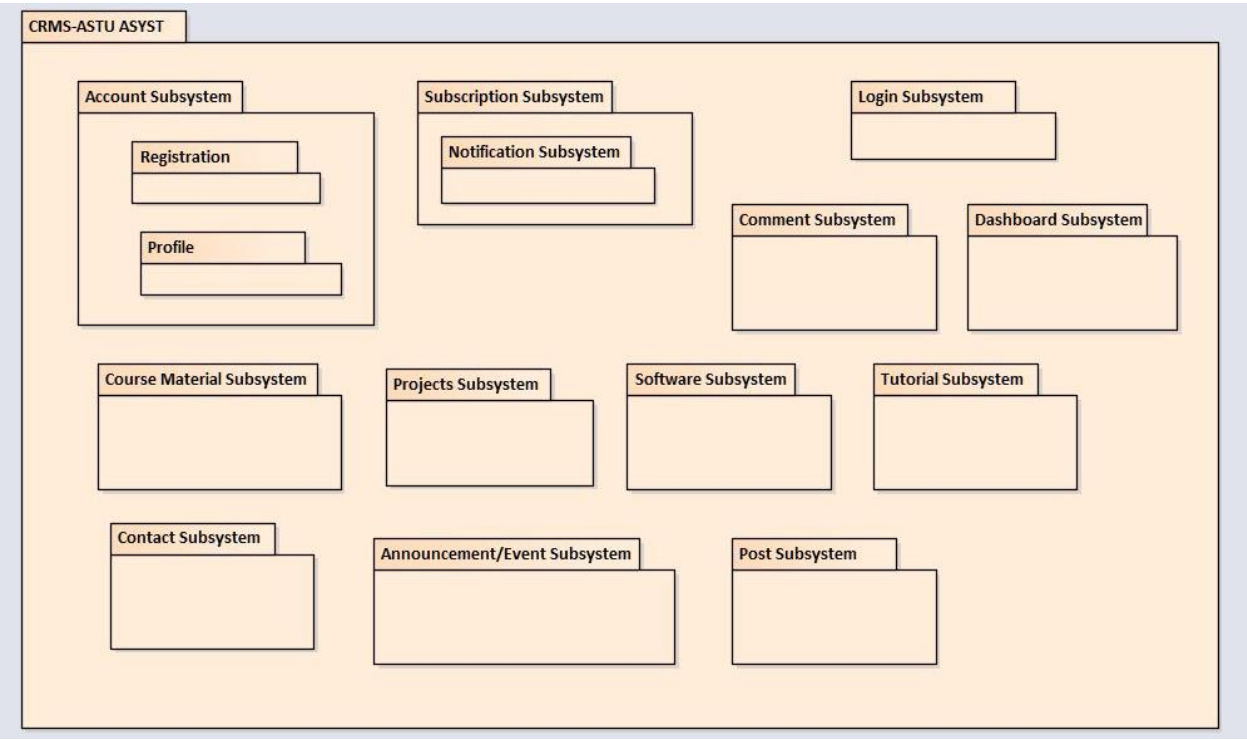

Figure 4.8 Subsystem Decomposition (Package Format)

4.3.3 Hardware Software Mapping

The system will rely on both hardware and software capabilities to perform efficiently; the computer can access the website through the internet so the computer should be equipped with network adapter, network drivers, web browsers and flash player. The webserver should run on a certain engine software like nginx or ngrok with Apache Database Client running PHP Maria DB or any other Database Client.

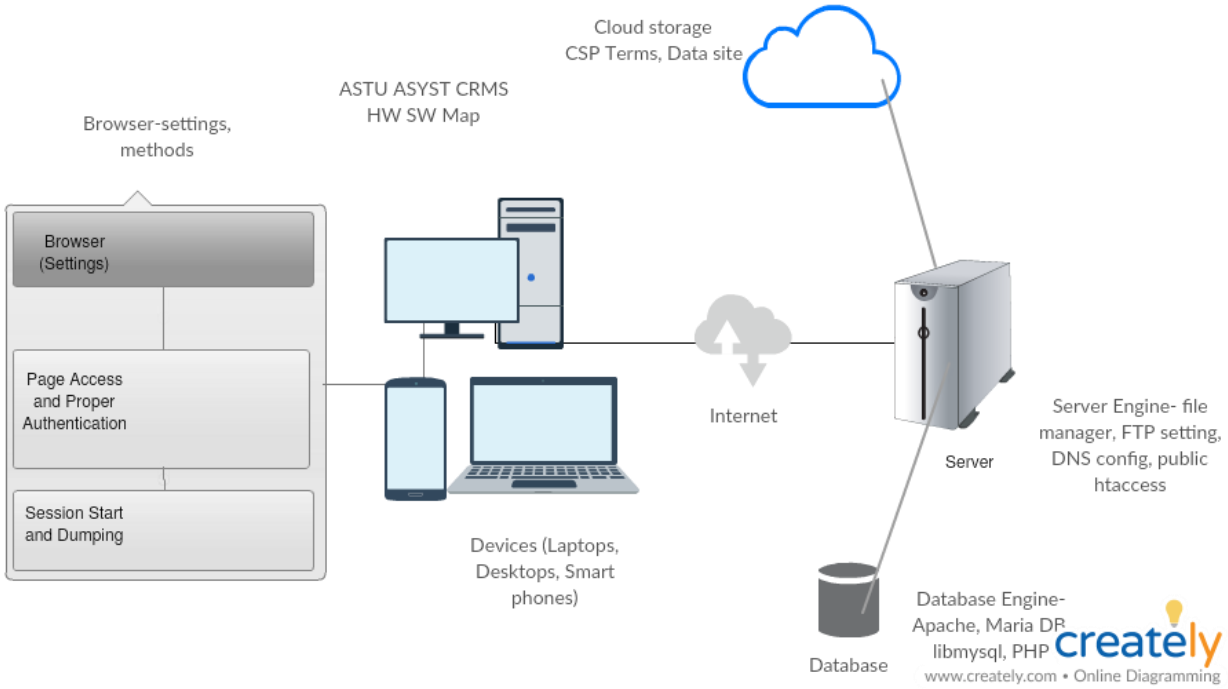

Figure 4.9 Hardware Software mapping

4.3.4 Persistent Data Management

This section provides a mapping of the class diagram's classes and objects that were identified in the requirement analysis phase into a relational database format.

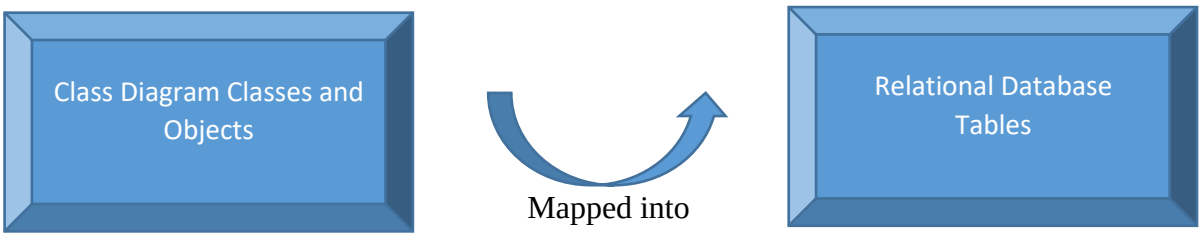

Figure 4.10 Persistent Data Managment

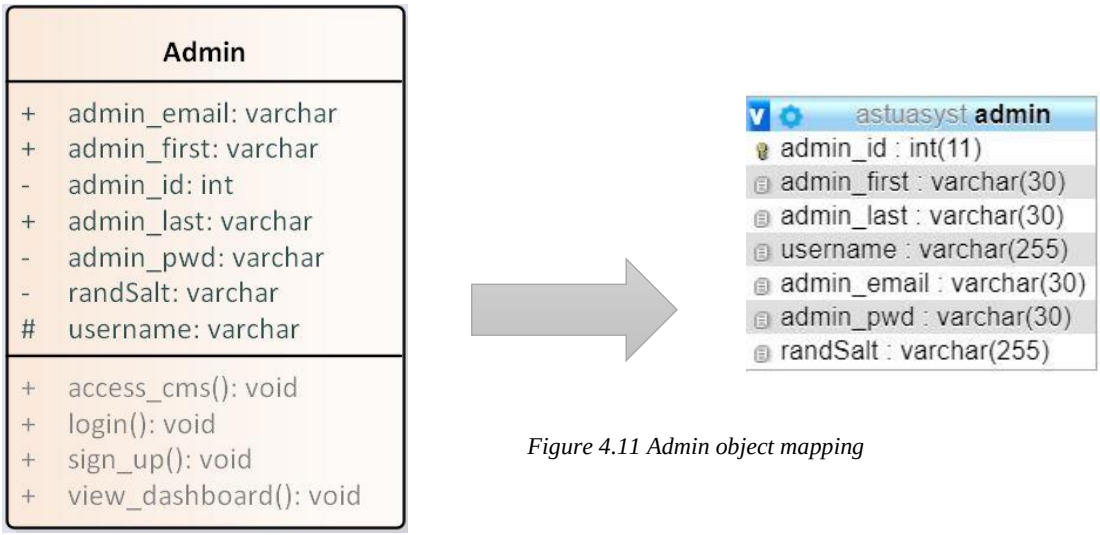

Figure 4.11 Admin object mapping

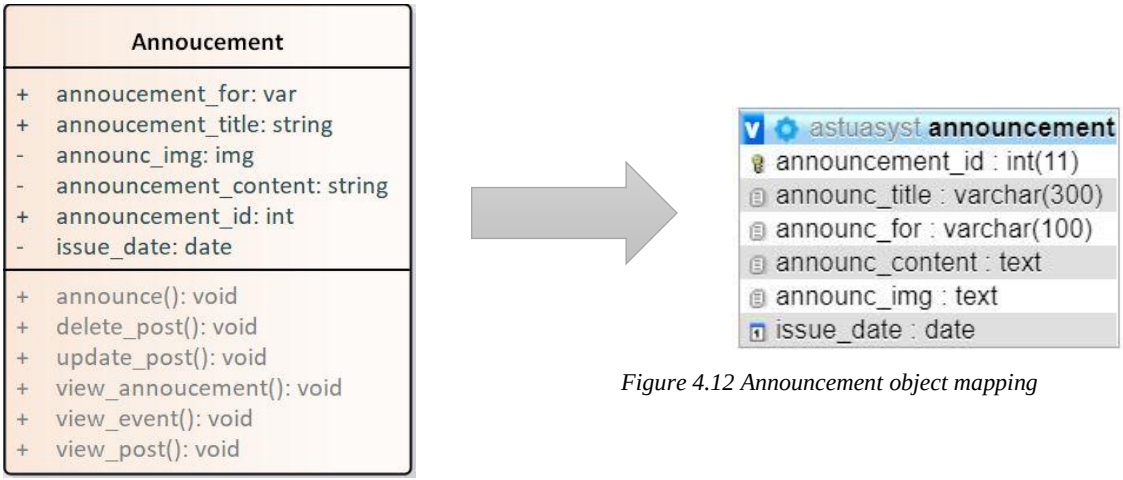

Figure 4.12 Announcement object mapping

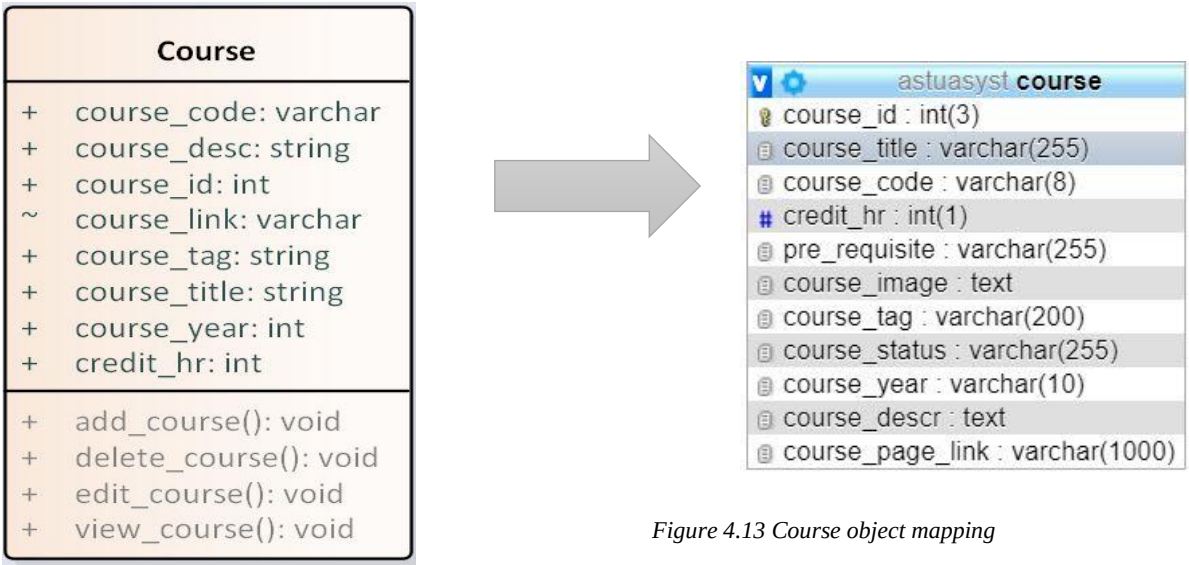

Figure 4.13 Course object mapping

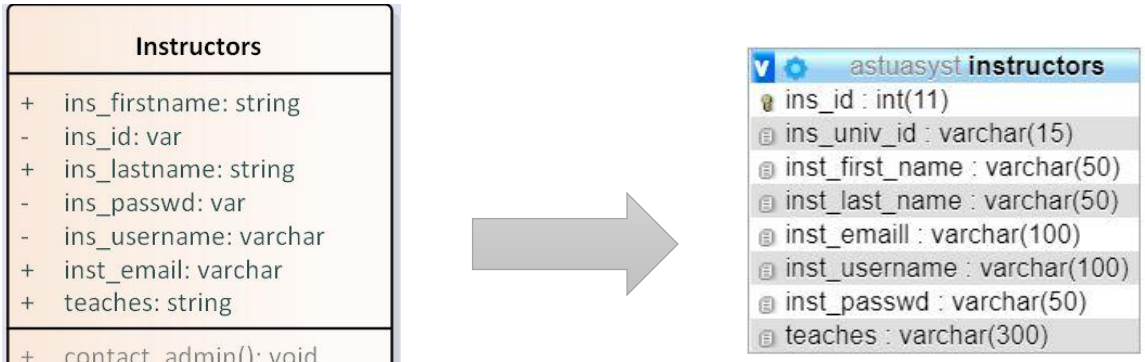

Figure 4.14 Instructor object mapping

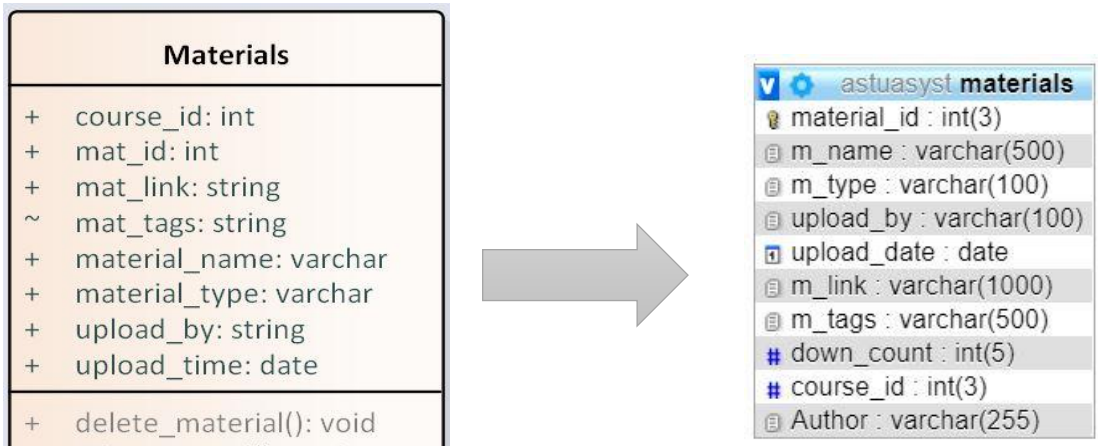

Figure 4.15 Material object mapping

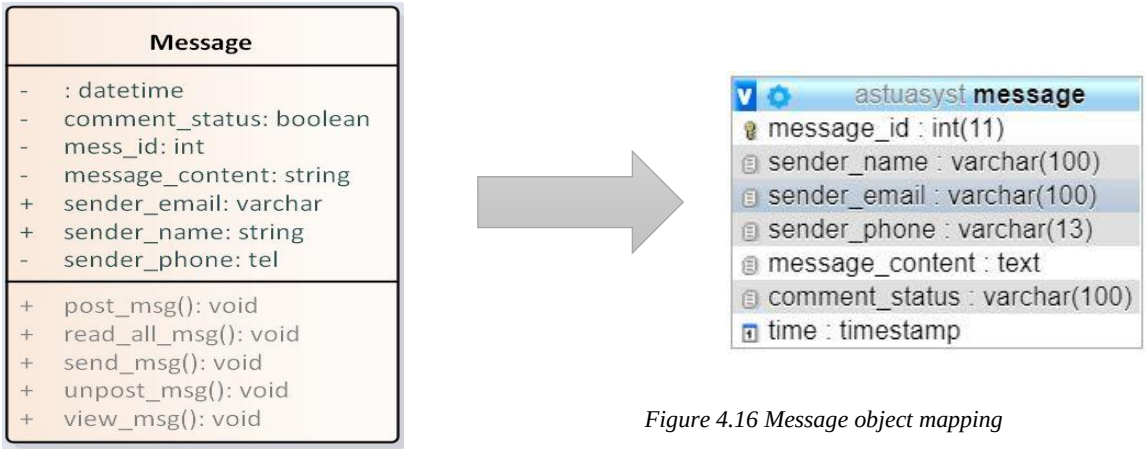

Figure 4.16 Message object mapping

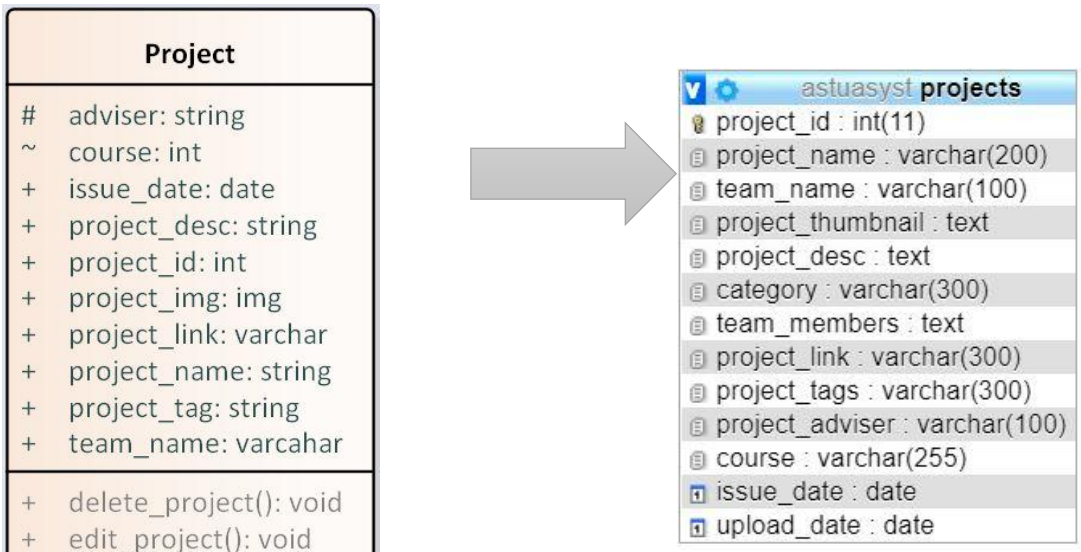

Figure 4.17 Project object mapping

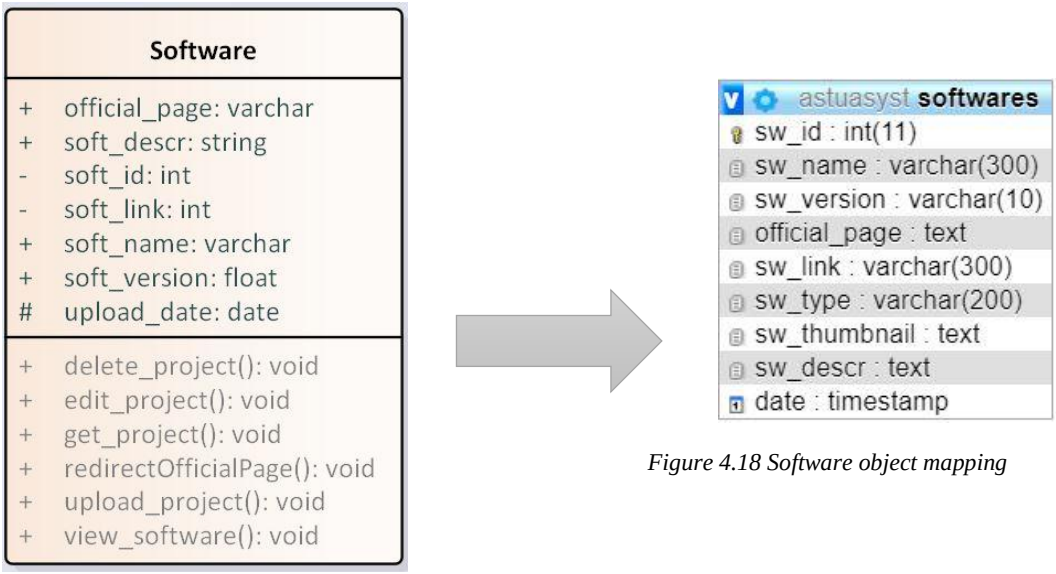

Figure 4.18 Software object mapping

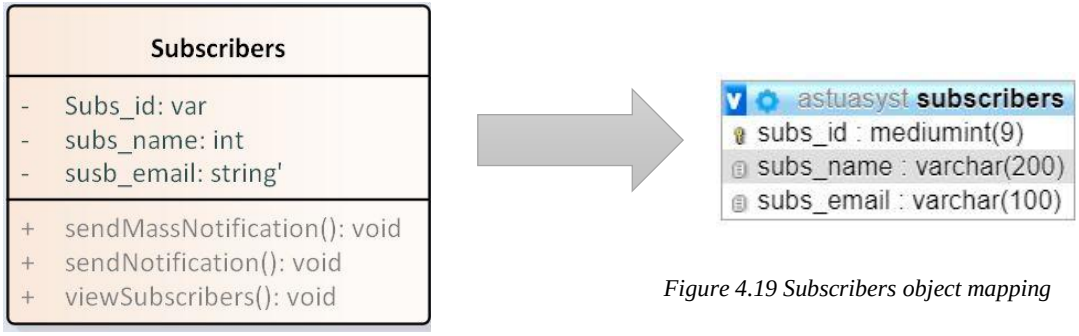

Figure 4.19 Subscribers object mapping

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 10.1.37-MariaDB - mariadb.org binary distribution
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.37 (Win32) OpenSSL/1.1.1a PHP/7.3.0
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: 401a40ebd5e281cf22215acdc170723a1519aaa9 $
- PHP extension: mysqli curl mbstring
- PHP version: 7.3.0

Component Diagram

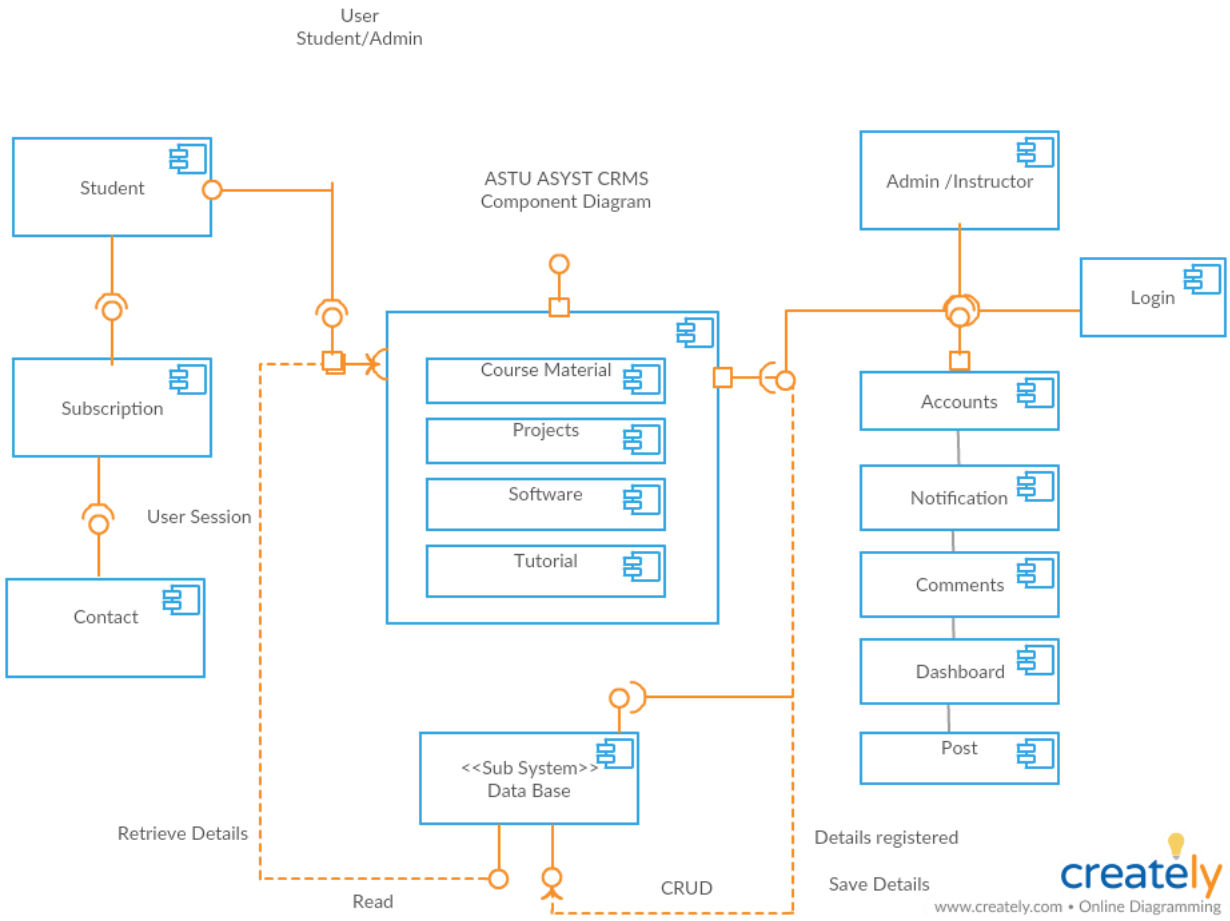

Figure 4.20 Component Diagram

Deployment Diagram

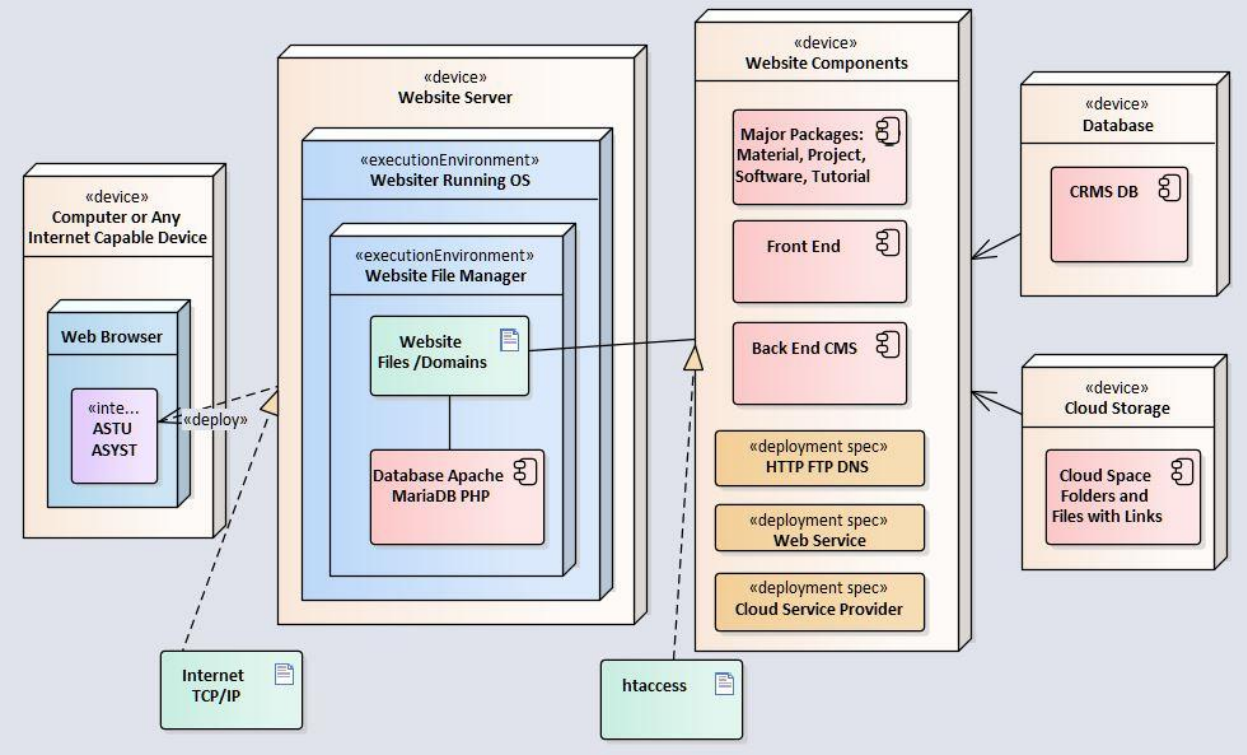

Figure 4.21 Deployment Diagram

4.3.5 Access Control and Security

Due to the prototype demo nature of the CRMS project, the access control and security issue is addressed in a testing level with necessary hash and encryption methods at security points like login subsystem. A crypt function of cost 12 is used from PHP built in function with a 22 character random salt.

A secure access control mechanism that prevents unauthorized or even malicious users from using the network must be incorporated. The access control mechanisms must operate fully transparent to any correspondent node in the network. End users of the access network should, apart from the initial authentication at login time, not notice any changes when using the network (i.e., standard applications must be fully supported). Network security is a broad topic that can be addressed at the data link, or media level at the network, or protocol, layer (the point at which Internet Protocol (IP) packets and routing updates are controlled), and at the application layer (where, for example, host-level bugs become issues).

Due to the limitation of the current working environment, the IPv6 will not be used in this CRMS project, rather in the future version.

Access Matrix

Actors / Object	Account	Course Material	Project	Software	Tutorial
Students		ViewMaterials() GetMaterials()	ViewProjects() GetProjects()	ViewSoftware() GetSoftware() RedirectOfficialPage()	ViewTutorial() GetTutorial()
Instructors	CreateAccount() ViewProfile() EditProfile()	ViewMaterial() GetMaterial() PostMaterial() EditMaterial() DeleteMaterial()	ViewProjects() GetProject() PostProjects() EditProject() DeleteProject()	ViewSoftware() GetSoftware() RedirectOfficialPage() PostSoftware() EditSoftware() DeleteSoftware()	ViewTutorials() GetTutorial()
Administrators	CreateAccounts() ViewAllProfiles() EditAllProfiles() DeleteAllProfiles()	ViewMaterials(*) GetMaterial() PostMaterials(*) EditMaterials(*) DeleteMaterials(*)	ViewProjects() GetProject() PostProjects(*) EditProject(*) DeleteProject(*)	ViewSoftware() GetSoftware() RedirectOfficialPage() PostSoftwares(*) EditSoftwares(*) DeleteSoftwares(*)	ViewTutorial() GetTutorial() PostTutorial() EditTutorial() DeleteTutorial()
Actors / Object	**Login**	**Subscription**	**Contact**	**Announcement/Post**	**Dashboard**
Students		Subscribe() GetNotification()	SendFeedback() Comment() ViewComments()	ViewAnnouncement() ViewPost() ViewEvent()	
Instructors	LogIn() LogOut() ForgotPassword() AccessCMS()	Subscribe() GetNotification()	SendFeedback() Comment() ViewComments() ContactAdmin()	ViewAnnouncement() ViewPost() ViewEvent()	ViewDashboard()
Administrators	LogIn() LogOut() ForgotPassword() UsersOnline() AccessCMS(*)	ViewSubscription() SendNotification() SendBulkNotification()	ViewComments(*) PostComments() ReplyMessage()	Announce() Post() UpdatePost() DeletePost()	ViewDashboard() Settings() Status()

Table 4.1 Access Matrix

4.3.6 Global Software Control

Global software control describes how the global software control is implemented. In particular, this section should describe how requests are initiated and how subsystems synchronize. This section should list and address synchronization and concurrency issues.

The following section will describe the software control implementation.

External control flow (between subsystems)
• Control flow is distributed within the CRMS system. This means that there is no central control instance.
• Each service has its own control flow with PHP page constraint, cookie assignment and session callback.
• Services request input (”needs”), wait for it and resume control when it arrives.
• The services use asynchronous callbacks to communicate with each other.
Concurrent control
The CRMS content management system handles communication between single services and balances abilities and needs of the services. Failure of one service must not affect the service manager and the other services.
• Multithreading
the service manager uses threads, so that a large number of services is able to use the service manager simultaneously. The service manager also handles asynchronous events within the system.
Internal control (within a single process)
• Avoiding Deadlocks

Processes must not callback a running service; in order to maintain a dead-lock free environment.
• Worker Threads for Each Service
each service has an own thread for communication, which communicates with the service manager and other services.

4.3.7 Boundary Conditions

Boundary conditions describes the start-up, shutdown, and error behavior of the system. One of the main proposals of Boundary Conditions is to describe how the whole system can be started, what services have to be initializes first.

Startup

The system in this case the website starts with composer JSON trigger of index page providing the home page with navigation to other pages. The subsidiary pages would round up to the “href” reference and get redirected accordingly.

The instructor and administrator subdomains also must initialize as the privileged users need to access the backend CMS at any given time with session configurations.

Termination

As primary users close the website's tab, the system returns to halt for that particular request; yet hold an active state for next website call.

As administrators and instructors logout of the backend CMS, the system directs the browser to dump all the sessions to null and release all cookies to cache.

Failure

As a non-existing link is entered in the system, the system routes to the pre-set 404 page using the .htaccess ERROR DOCUMENT. The PHP and html tags are routed to a shorter URL for security purposes. In case of connection error or Server failure provide an interruption page for the users to visit the website later on.

Error	Cause	Forecast Solution
Website GUI not working	Could be many internal causes	Try again later or use a plugin.
Some fatal errors	Malware	Scan with malware scanner
Internet not working	Improperly configured APN settings. No internet from the user internet service provider (ISP).	Properly configure internet connection settings as well as proxy if necessary.
Server Not found	Internet connection fail	Make sure Network connection or firewall (proxy) protection settings are correct.

Table 4.2 Boundary Condition

4.4 Subsystem Services

This section describes the services each subsystem provides along with their needs and abilities.

Subsystem	Services	Class | Method |Table
Account Subsystem	To register instructors and set up a user account and profile for instructors which would give them access to the CMS.	User Users Online Instructors Create,Update,Delete
LogIn Subsystem	Using the created profile, allow users to authenticate into the backend CMS.	Users Users Online Session
Course Material Subsystem	It offers the materials in an organised and interactive view in the front end and give upload option fro the CMS.	Materials Users Instructors Create,Read,Update,Delete
Projects Subsystem	Holds the project resources with download option and	Project Users

	allow authenticated users to upload from the CMS.	Instructors Create,Read,Update,Delete
Software Subsystem	Holds the software resources with download & Official page option and allow authenticated users to upload from the CMS.	Software Users Instructors Create,Read,Update,Delete
Tutorial Subsystem	Offers the tutoial resources-videos, files, books, classroom option and allow admins to upload from the CMS.	Tutorial Users Admins Create,Read,Update,Delete
Subscription Subsystem	Students can subscribe for a newsletter and they can be notified when new material is uploaded to the website.	Subscribers Admins Notify
Contact Subsystem	Website users can leave a comment, give feedbacka and ask questions from the contact page and admins can view this messages and respond accordingly.	Message Send() Post()
Announcement/Event Subsystem	Administrators can post announcements and post events to the website.	Announcement Event Create,Read,Update,Delete
Post Subsystem	Admins can post news, current issues and recommendations to the homepage of the website if possible daily to boost interactivity.	Posts Admins Users Create,Read,Update,Delete
Comment Subsystem	Admins can select out the messages and comments received and post them to a feedback section on the website.	Comments Admins Users Create,Read,Update,Delete
Dashboard Subsystem	The CMS would provide a general status of aspects included in the website with graphical and tabular format.	Dashbaord Admins Instructors View()

Table 4.3 Subsystem Services

Glossary Wireframe Interface Design of Front and Backend of the system

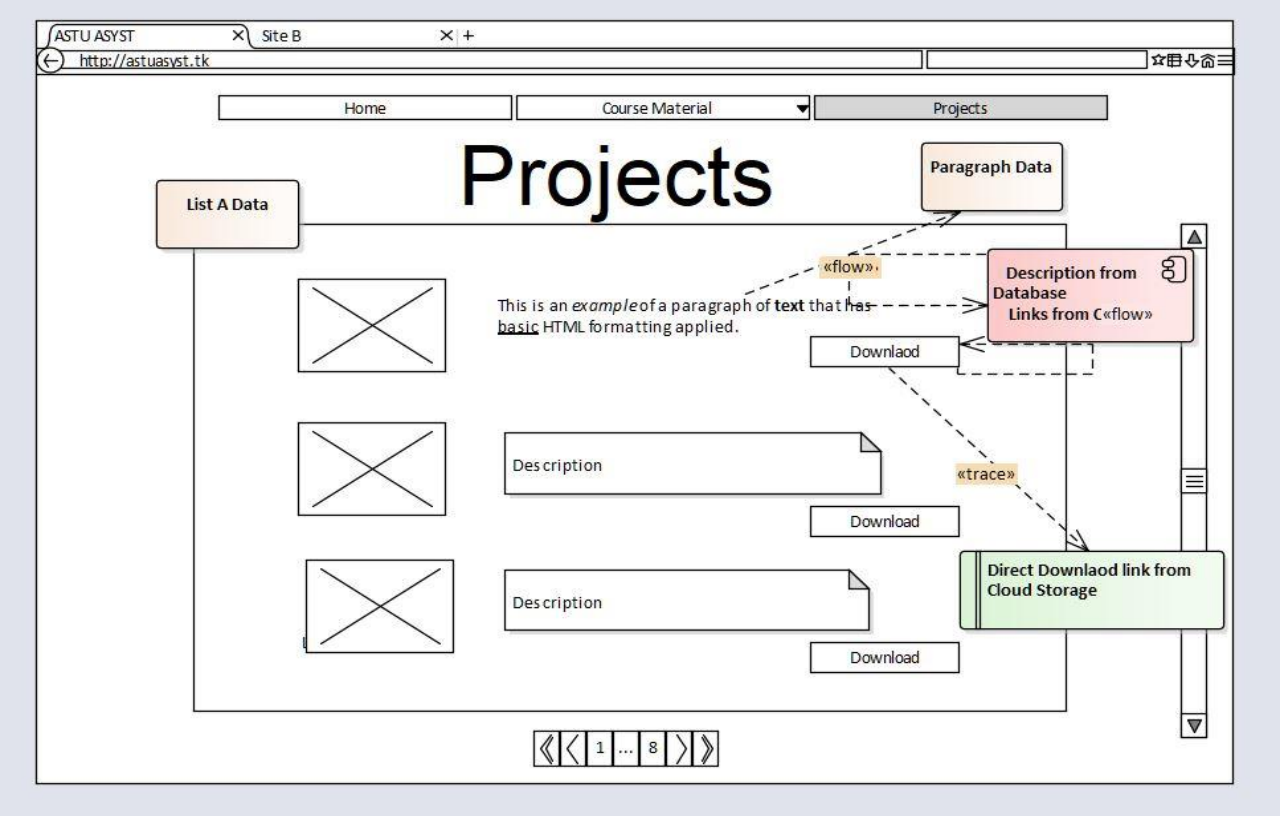

Figure 4.22 Wireframe Front End Skeletal View

Figure 4.23 Wireframe Back End Skeletal view