Table of Contents

ACKNOWLEDGEMENT

First and foremost we thank the almighty, the greatest Electronics and Communication Engineers of the universe for giving us such a speculate years.

This is the time to express foremost thanks and gratitude to the honorable **Shri. R. KANDASAMY**, Chairman of Muthayammal Educational Trust and Research Foundation for providing the excellent infrastructure to carry out this internship.

We wish to express our heartfelt gratitude and hearted thanks to our Secretary **Dr. K. GUNASEKARAN, M.E., Ph.D., F.I.E,** Muthayammal Educational Trust and Research Foundation, for the motivation, esteemed co-operation and encouragement throughout this internship.

We express our sincere thanks and profound gratitude to the principal **Dr. M. MADHESWARAN, M.E., Ph.D., MBA,** Muthayammal Engineering College, for granting permission to undertake this internship.

Perhaps our deepest thanks go to **Dr. U. SARAVANAKUMAR, M.E., Ph.D.,** Head of the Department, Department of Electronics and Communication Engineering, for his stimulating comments, which helped us in bringing things in our way.

We also express gratitude and heart full thanks to our Internship coordinator **Mr. S. BHOOPALAN M.E., (Ph.D).,** Associate professor, who has motivated us in the completion of the internship.

We are very thankful to **Mrs. P. SUBHASUNDARI M.E.,** Supervisor, our Internship guide, who has constantly motivated and inspired us in rending to completion of the Internship. We also express our gratefulness to our parents, our faculty members and friends for their affectionate blessings and loving co-operation at all stages of this academic venture.

ABOUT THE ORGANIZATION

Zuci is a digital organization focused on the craft of building software which we have perfected over the years. A perfect blend of design thinking, engineering perfection, and customer centricity in our DNA has enabled us to help small, medium and large organizations with superior digital transformation solutions.

Vasudevan Swaminathan, Venkatesh Veerachamy and Anil Kumar, the founders of Zuci Systems were working in technological roles for over 15 years. Their love for technology and quest for tougher challenges, was the genesis of Zuci Systems. The founders share similar values, trust and pursuit for excellence.

Zuci, like any other startup, faced challenges in its initial stages. The seed funds to start and maintain the company had faced many roadblocks. Structuring the organization was an aspect of the business that none of the founders had any idea about. Letting prospective clients know that they were good enough to handle their technology needs was a struggle.

They have been able to consistently create Zen moments for our clients only because of the culture that everyone associated with Zuci has contributed to from day one. Our DNA of delivering excellence was powered by it, long before we even coined a word for it.

ABOUT THE INTERNSHIP

In this internship we have undergone the dotnet full stack training for the developer modules. During this period we have gone through the contents like C# programming language, SQL, Entity framework core, API and MVC.

C# (C sharp):

C# (pronounced "See Sharp") is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust applications that run in .NET. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers.

C# is an object-oriented, component-oriented programming language. C# provides language constructs to directly support these concepts, making C# a natural language in which to create and use software components. Since its origin, C# has added features to support new workloads and emerging software design practices. At its core, C# is an object-oriented language.

Several C# features help create robust and durable applications. Garbage collection automatically reclaims memory occupied by unreachable unused objects. Nullable types guard against variables that don't refer to allocated objects. Exception handling provides a structured and extensible approach to error detection and recovery. Lambda expressions support functional programming techniques. Language Integrated Query (LINQ) syntax creates a common pattern for working with data from any source.

SQL:

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc. This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDSMS.

The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database. SQL allows the data professionals and users to retrieve the data from the relational database management systems.

When we are executing the command of SQL on any Relational database management system, then the system automatically finds the best routine to carry out our request, and the SQL engine determines how to interpret that particular command.

Entity Framework Core:

Entity Framework Core is the new version of Entity Framework after EF 6.x. It is open-source, lightweight, extensible and a cross-platform version of Entity Framework data access technology.

Entity Framework is an Object/Relational Mapping (O/RM) framework. It is an enhancement to ADO.NET that gives developers an automated mechanism for accessing & storing the data in the database.

EF Core supports two development approaches 1) Code-First 2) Database-First. EF Core mainly targets the code-first approach and provides little support for the database-first approach because the visual designer or wizard for DB model is not supported as of EF Core.

In the code-first approach, EF Core API creates the database and tables using migration based on the conventions and configuration provided in your domain classes. This approach is useful in Domain Driven Design (DDD).

In the database-first approach, EF Core API creates the domain and context classes based on your existing database using EF Core commands. This has limited support in EF Core as it does not support visual designer or wizard.

API:

An API, or application programming interface, is a set of rules or protocols that let software applications communicate with each other to exchange data, features and functionality.

APIs simplify application development by allowing developers to integrate data, services and capabilities from other applications, instead of developing them from scratch. APIs also give application owners a simple, secure way to make their application data and functionality available to internal departments within their organizations. Application owners can also share or market that data and functionality to business partners or third parties.

APIs simplify design and development of new applications and services, and integration and management of existing ones. But they offer other significant benefits to developers and organizations at large.

Web API is a programming interface/application type that provides communication or interaction between software applications. Web API is often used to provide an interface for websites and client applications to have data access. Web APIs can be used to access data from a database and save data back to the database.

ASP.NET Web API is a framework that make it easy to build HTTP web service that reaches a bored range of clients, including browser, mobile applications, Desktop application and IOTs.

The following HTTP methods are most commonly used in a REST based architecture.

1. GET − Provides a read only access to a resource.
2. PUT − Used to create a new resource.
3. DELETE − Used to remove a resource.
4. POST − Used to update an existing resource or create a new resource.

ASP.NET Web API:

Following are some benefits of working with an ASP.NET Web API:

- It works the way HTTP works, using standard HTTP verbs like GET, POST, PUT, DELETE for all CRUD operations.
- Full support for routing.
- Response is generated in JSON and XML format using MediaTypeFormatter.
- It can be hosted on IIS as well as auto-hosted outside of IIS.
- Supports template binding and validation.
- Supports URL patterns and HTTP methods.
- It has a simple form of dependency injection.
- Can be versioned.

MVC (Model View Controller):

MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

Using the MVC pattern for websites, requests are routed to a Controller that is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display and provides it with the Model. The View renders the final page, based on the data in the Model.

Create clean model classes and easily bind them to your database. Declaratively define validation rules, using C# attributes, which are applied on the client and server.

ASP.NET supports many database engines including SQLite, SQL Server, MySQL, PostgreSQL, DB2 and more, as well as non-relational stores such as MongoDB, Redis, and Azure Cosmos DB.

Simply route requests to controller actions, implemented as normal C# methods. Data from the request path, query string, and request body are automatically bound to method parameters.

The Razor syntax provides a simple, clean, and lightweight way to render HTML content based on your view. Razor lets you render a page using C#, producing fully HTML5 compliant web pages.

OUTCOMES

- Build a web application using ASP.NET MVC, integrating with a SQL database for data storage and retrieval. Implement CRUD operations and user authentication to manage application access securely.
- Create a RESTful API using ASP.NET Core Web API, enabling communication between different software components. Implement endpoints for various resources and ensure proper documentation using tools like Swagger.
- Design and optimize a SQL database schema, focusing on performance tuning techniques such as indexing and query optimization. Utilize Entity Framework Core for seamless data access within your .NET application.
- Integrate frontend, backend, and database components to ensure smooth communication and functionality. Write unit tests and integration tests using testing frameworks like NUnit to validate the behavior of your application.
- Implement security measures such as authentication and authorization mechanisms in your web application and API. Utilize JWT tokens or OAuth for user authentication and authorization policies to control access to resources.