TRIBHUVAN UNIVERSITY

INSTITUTE OF COMPUTER STUDIES

NATIONAL COLLEGE OF COMPUTER STUDIES

AN INTERNSHIP REPORT PROPOSAL

ON

CLOUD DATA ENGINEER

AT

PMSQUARE NEPAL

SUBMITTED BY:

PRATIK DHAKAL [23842/076]

SUBMITTED TO:

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

26TH MAY, 2024

Acknowledgments

First and foremost, I would like to extend my sincere appreciation and gratitude to PMsquare Nepal for giving me the invaluable opportunity to intern with their esteemed organization. Without their acceptance and support, this proposal report would not have been possible. I am truly grateful for the experiences and knowledge gained during my internship with them.

I also express my sincere thanks to Tribhuvan University and the Institute of Science and Technology (IOST) department for incorporating the internship program into the B.Sc. CSIT curriculum. This initiative provided me with the motivation and platform to explore my field of interest in a practical, real-world context.

Lastly, I am deeply appreciative of our project coordinator, Mr. Rajan Poudel. His professional guidance and support have been immensely valuable throughout this journey. His advice and direction have greatly contributed to the completion of this proposal report.

Thank you all for your unwavering support and encouragement.

Sincerely,
Pratik Dhakal
Symbol No: 23842/076
NCCS College

Contents

List of Figures

List of Abbreviations

APIs	Application Programming Interfaces
AWS	Amazon Web Services
BI	Business Intelligence
CCPA	California Consumer Privacy Act
CSS	Cascading Style Sheets
CTEs	Common Table Expressions
DBMS	Database Management System
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DQL	Data Query Language
ELT	Extract, Load, Transform
ETL	Extract, Transform, Load
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
IoT	Internet of Things
JSON	JavaScript Object Notation
ML	Machine Learning
NCCS	National College of Computer Studies
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SVG	Scalable Vector Graphics
TCL	Transaction Control Language
TSP	Traveling Salesman Problem

1. Introduction

I am currently engaged in an internship at PMsquare Company as a Cloud Data Engineer, as part of the CSIT curriculum at NCCS College. This internship presents a valuable opportunity to apply theoretical knowledge in cloud computing and data engineering to real-world scenarios within a cloud-focused organization. The project is designed to enhance my practical skills in cloud technologies and contribute to PMsquare's expertise in cloud-based data solutions. Additionally, I am exploring other fields, particularly frontend development, to broaden my skill set and understanding of different aspects of technology.

PMsquare Company is a leading provider in cloud-based data analytics and business intelligence solutions, specializing in leveraging cloud technologies to drive innovation and efficiency. With a strong emphasis on cloud computing services, PMsquare is at the forefront of transforming data into actionable insights through scalable and secure cloud platforms. As an intern focusing on cloud data engineering and exploring frontend development, my role is to contribute to the company's cloud initiatives by implementing data engineering solutions that optimize data processing and analysis in the cloud environment, while also gaining insights into frontend technologies.

This internship project proposal outlines my objectives, timeline, and expected outcomes for the one-month project at PMsquare Company, aligning with the cloud-centric approach of the company and my role as a Cloud Data Engineer intern. It provides a structured plan for my supervisor at NCCS College to review and approve for my internship documentation, emphasizing the practical application of cloud technologies in a professional setting and my exploration of frontend development to enhance my overall skill set.

2. Problem Statement

PMSquare Nepal is offering a three-month internship for a Cloud Data Engineer to contribute to the development of a cloud-based data architecture for our e-commerce platform. The project aims to address the challenges of managing vast amounts of data from various sources, including customer transactions, website interactions, inventory systems, and external APIs.

The intern's primary focus will be on designing and implementing a scalable, efficient, and secure data pipeline that supports real-time analytics, enhances data accessibility, and ensures data integrity and security. This involves creating systems for efficient data ingestion, integrating diverse data sources, and selecting suitable storage solutions like data lakes and warehouses.

In addition to the main project, the intern will have access to an AWS Cloud Practitioner course to develop skills in using AWS cloud services effectively. This course will provide foundational knowledge and practical experience in utilizing AWS services, enhancing the intern's ability to design and implement cloud-based solutions.

The intern will also work on developing ETL/ELT processes for data cleaning, transformation, and loading, catering to both real-time and batch processing requirements. Ensuring compliance with data privacy regulations such as GDPR and CCPA, implementing security measures, and enabling efficient data retrieval and analysis will be key responsibilities.

This role requires familiarity with cloud platforms like AWS, Azure, or Google Cloud, as well as proficiency in tools such as Apache Spark and Hadoop. Knowledge of security best practices and programming languages like Python, Java, or Scala is essential.

3. Objectives

The major objectives of the cloud data engineering intern are:

- Continuously learn and explore emerging cloud technologies and trends to stay current in the rapidly evolving field of cloud data engineering.
- Developing skills in cloud technologies, data engineering, and front-end development.
- Contributing to Real-World Projects: Actively engage in the development and implementation of real-world cloud data engineering projects, gaining hands-on experience and practical insights into industry best practices.

The intern is expected to remain updated with emerging cloud technologies and trends, ensuring relevance in the dynamic field of cloud data engineering. By proactively seeking out new knowledge and exploring innovative solutions, the intern can contribute to the organization's growth and competitiveness.

In addition to learning, the intern is tasked with developing proficiency in various areas including cloud technologies, data engineering, and front-end development. Through hands-on practice and guided learning, the intern can build a strong foundation of skills essential for success in the field.

The intern is expected to actively contribute to the development and implementation of real-world cloud data engineering projects. By working on live projects, the intern gains valuable experience and practical insights into industry best practices. This hands-on approach fosters a deeper understanding of concepts and prepares the intern for future challenges in the field.

4. Description of Internship Work

4.1 HTML and CSS Essentials

The internship work covered the fundamental building blocks of web development, starting with HTML. Key topics included: Anchor tag (<a>) and its attributes for creating hyperlinks, controlling link destinations, and styling link states Specialized HTML tags like <pre>, <bdo>, <map>, <picture>, and <iframe> for advanced functionality The importance of semantic HTML and its impact on SEO optimization Exploring the ¡form¿ tag and its key attributes for form submission The internship also delved into CSS, covering: Understanding the cascade, specificity, and the !important declaration Inheritable and non-inheritable CSS properties

4.1.1 Flexbox and Grid Layouts

Flexbox and Grid are two powerful CSS layout modules that provide flexible and responsive ways to arrange elements on a web page. Flexbox is a one-dimensional layout system that works by distributing space along a single axis, either horizontally or vertically. It allows for easy control over the size, position, and alignment of elements within a container.
Grid Layout is a two-dimensional system that enables the creation of complex grid-based layouts. It allows developers to define rows and columns to precisely control the placement and size of elements. Grid provides a powerful way to dynamically adjust the number of columns based on the available space while maintaining a minimum column width. This results in a smooth and flexible layout across different screen sizes and resolutions.

4.1.2 Understanding the Box Model

The Box Model is a fundamental concept in CSS that defines the layout and sizing of elements on a web page. Every HTML element is considered a rectangular box, and the Box Model describes the dimensions of that box. The key components of the Box Model are:
Content: The actual content of the element, such as text or images Padding: The space between the content and the border Border: The line surrounding the padding and content Margin: The space between the border and the outside world Elements can have two types of display values:

Block: These elements break into a new line and occupy the available full width. Examples include `<div>`, `<p>`, and `<h1>`. Inline: These elements do not break into a new line and only occupy the space necessary for their content. Examples include `<a>`, `<span>`, `<em>`, and `<strong>`. Width and height properties are not respected for inline elements.
The Box Model has two variations: Standard Box Model: In this model, the width and height properties define the size of the content box. Padding and border are added to the dimensions of the content box to calculate the total size taken up by the box. The margin is not counted toward the actual size of the box.
Alternate Box Model: The alternate box model, often preferred in modern web development, is set by the property box-sizing: border-box. In this model, width and height properties define the actual space taken up by the box, including the content box, padding, and border. The margin is still calculated outside of the box.

4.2 Margin Collapsing

Margin collapsing is a behavior in CSS where the margins of adjacent elements collapse into a single margin. This typically occurs between block-level elements when no content or padding is separating them vertically. The collapsed margin is equal to the larger of the two margins, effectively combining them.

4.3 Advanced CSS Techniques

Building on the CSS foundations, the internship explored more advanced CSS concepts: CSS Animations using the transform and transition properties Leveraging the prefers-reduced-motion media query to respect user preferences The translate property for 3D transformations Crafting smooth transitions between CSS property values

4.3.1 CSS Animations: Transform and Transition

CSS provides two properties for creating animations: transform and transition.
The transform property allows for various transformations of elements, such as rotation, scaling, skewing, or translating them in a 2D or 3D space. The order of transformation functions within the transform property matters, as each function is applied sequentially.
The transition property facilitates the creation of smooth animations by gradually changing CSS properties over a specified duration. It enables developers to control the timing and easing of the animation, creating visually appealing effects.
The prefers-reduced-motion media query enables developers to respect users' preferences re-

garding animation. This query can control whether animations are enabled based on the user's system preferences. The translate property is a specific transformation function within transform that allows for independent translation in three dimensions compared to the two dimensions of the translate() function.

4.4 Git and Deployment

The internship work also covered version control and deployment, including: Git functionalities like Stash, Staging, Reset, and Rebase and Deploying web applications to Netlify

4.4.1 Git Stash

Git Stash serves as an ephemeral storage area, allowing developers to temporarily shelve changes that are not yet ready to be committed. It provides a way to set aside modifications, ensuring they can be reapplied or retrieved at a later time. The following commands are commonly used with stash:
git stash pop: Removes changes from the stash and reapplies them to the working copy.
git stash apply: Reapplies changes from the stash to the working copy, preserving them in the stash for future use across branches.

4.4.2 Staging in Git

Staging acts as an intermediate step between the working directory and the commit history. With staging, developers can group related changes before committing them, facilitating better organization and granularity in version control. Key aspects of staging include: git add: Promotes pending changes from the working directory to the staging area, preparing them for the next commit. Grouping Changes & Partial Commits: Staging allows for grouping changes not only between files but also within specific parts of files, enabling precise control over commit content. The git add -p command permits developers to select portions of files to include in the next commit, enhancing flexibility and granularity.
git commit –amend: Allows for editing the most recent commit instead of creating a new commit for subsequent changes, offering a convenient way to refine commit history.

4.4.3 Git Reset

Git reset is a powerful command that allows developers to manipulate the state of the working directory, staging index, and commit history. Understanding its different modes——hard,

–mixed, and –soft—is crucial for effectively managing changes in a Git repository.
The working directory represents the current state of files on the local filesystem. Changes to the working directory can be viewed using git status, with modified files prefixed with the ”modified” label. The staging index tracks changes that have been promoted with git add and are scheduled to be included in the next commit. These changes are displayed in green under the ”Changes to be committed” section of git status.
The commit history stores permanent snapshots of the repository state. Git commit adds changes to a permanent snapshot stored in the commit history.
The git reset command resets the HEAD pointer to the specified commit, modifying the staging index and working directory accordingly. The –hard mode updates the commit history ref pointers, staging index, and working directory to match the specified commit, discarding any pending changes. The –mixed mode updates the commit history ref pointers and resets the staging index, moving any undone changes from the staging index to the working directory. The –soft mode updates the commit history ref pointers but leaves the staging index and working directory untouched.

4.4.4 Rebasing in Git

Rebasing involves the process of moving a sequence of commits to a new base commit. It takes the commits in your current working branch and applies them to the head of the specified base branch, effectively incorporating changes from the new base into your branch's commit history.
Interactive rebasing allows you to alter individual commits during the rebase process. The git rebase -i ¡base¿ command opens an interactive rebase editor where you can pick, edit, squash, or drop commits as needed.

4.4.5 Deployment on Netlify

Deploying a website on Netlify can be done by connecting a GitHub repository or uploading the build HTML file. It offers several implications and functionalities, almost resembling a Continuous Integration and Continuous Deployment (CI/CD) pipeline albeit only for static content.
Some of the useful features of Netlify include:
Automatic Deployment: When a GitHub repository is connected to Netlify, any changes pushed to the repository trigger an automatic deployment process on Netlify.

Build Process: Netlify automatically builds the website based on the files and configurations present in the repository.
Preview Deployments: Netlify creates preview deployments for each pull request on the GitHub repository, allowing developers to preview changes before merging them into the main branch.

4.5 React and TypeScript

4.5.1 React

Conditional Rendering of UI

Use the && operator to conditionally render elements Don't put numbers on the left side of &&, as 0 will be rendered instead of nothing.Fix by ensuring the left side is a boolean, e.g. messageCount > 0 && New messages .

Keys in React

Keys must be unique among siblings, but can be reused across different arrays Keys must be stable identifiers associated with each item in a list Avoid using the index of an array as a key, as it can lead to unintended behavior when inserting or deleting items State as a Snapshot In React, state represents a snapshot of the component at a particular moment in time When a component re-renders, it calculates the JSX based on the current state snapshot To update the same state multiple times before the next render, pass a function to setState that calculates the next state based on the previous one

Immutability

Props and state should never be mutated directly Copy objects and arrays, update the copied object, then update state using the setter function Managing and Structuring State Group related state variables together in the same state object Avoid contradictory state representations Calculate derived data dynamically during rendering instead of storing it in state Mirror props in state only when necessary to track changes internally Ensure each piece of data is stored in only one location to avoid duplication Keep the state tree as shallow as possible to avoid complexity Using Set instead of Array When dealing with large arrays and ensuring uniqueness, using Set can provide better performance than Array.includes()

Effects

Use the useEffect hook to declare an effect, specifying dependencies and an optional cleanup function Effects are not needed for tasks like transforming data for rendering or handling user events

4.5.2 TypeScript

Interface vs Type

- Prefer using `interface` for defining object shapes.
- `interface` allows extending and implementing, suitable for defining contracts.

Compiler Options

- `noImplicitAny` enforces explicit typing to avoid implicitly inferred types as `any`.
- `strictNullChecks` enhances handling of `null` and `undefined`.

Union Types

- Union types result in an intersection of properties from the constituent types.

Type Notations

- `String[]`: Equivalent to `Array<string>`.
- `Record<string, string>`: Defines an object type with string keys and string values.

ComponentProps

- Used to extract props from a custom component using `ComponentProps<typeof Component>`.
- Facilitates type inference when a third-party component doesn't export its props types.

Overriding Default Props

- Can override default props of HTML elements by using `Omit` to remove specific properties from the HTML element type definition, followed by an intersection with the new type definition.

Discriminated Unions

- Allows representing different states such that only possible states are defined.
- Overcomes the usage of optional types while representing different states.

4.6 Cloud Computing Fundamentals

4.6.1 Brief History of Cloud Computing

The origin of networked computing in the 1960s and 1970s laid the groundwork for the paradigm of cloud computing. During this period, the development of time-sharing systems allowed multiple users to access a centralized computer simultaneously, sharing its resources remotely.

The rise of the internet in the late 20th century further catalyzed the development of distributed computing and laid the foundation for cloud computing. In 1999, Salesforce emerged as a notable example of successful cloud computing adoption, pioneering the idea of delivering software programs over the Internet.[1]

Since the early 2000s, Amazon has played a pivotal role in shaping the modern cloud computing landscape. In 2006, Amazon Web Services (AWS) launched its Elastic Compute Cloud (EC2) service, providing scalable compute capacity in the cloud. AWS's innovative approach revolutionized the IT industry and paved the way for other cloud providers to follow suit.

Today, AWS, along with competitors like Google Cloud Platform (GCP), Microsoft Azure, IBM Cloud, and others, has become a dominant force in the cloud computing market. These cloud providers offer a wide range of services and solutions, enabling organizations to leverage cloud computing for increased scalability, flexibility, and efficiency.

4.6.2 Cloud Deployment Models

There are three main cloud deployment models:

- **Public Cloud**: Public clouds are owned and operated by third-party cloud service providers, which deliver computing resources like servers, storage, and networking over the Internet. Public clouds can save companies from the expensive costs of purchasing, managing, and maintaining on-premises hardware and application infrastructure.
- **Private Cloud**: A private cloud refers to cloud computing resources used exclusively

by an organization. While it provides greater control and customization over the hardware and software, some organizations may find it difficult and expensive to manage the infrastructure.

- **Hybrid Cloud**: Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. Hybrid cloud models are popular for organizations that have to meet stringent data protection and compliance requirements.

4.6.3 Cloud Service Models

Cloud Computing can provide different services based on levels of abstraction and management responsibilities:

- **IaaS (Infrastructure as a Service)**: IaaS abstracts the underlying hardware infrastructure, including compute, networking, firewalls, security, data centers, and maintenance. It provides storage, backup, recovery, and high-performance computing capabilities.
- **PaaS (Platform as a Service)**: PaaS provides a complete development and deployment environment, combining IaaS with middleware, development tools, business intelligence (BI) services, and database management systems (DBMS). It accelerates application development and enables real-time data processing.
- **SaaS (Software as a Service)**: SaaS delivers cloud-based applications over the internet, allowing users to rent the use of software and access it remotely via web browsers or APIs. It streamlines software delivery and management through multi-tenancy.

4.6.4 Shared Responsibility Model

The Shared Responsibility Model is a framework that delineates the responsibilities for securing different aspects of the cloud computing environment between the cloud service provider and the customer.[2] The division of responsibility is based on the type of service used, with the responsibility of the consumer decreasing as one goes from IaaS services to SaaS services.

- For **IaaS**, the customer is responsible for managing the operating system, applications, data, runtime, middleware, and network controls. The cloud provider is responsible for securing the physical infrastructure, virtualization layer, and facilities.
- In **PaaS**, the customer is responsible for managing the applications and data. The cloud

provider is responsible for securing the runtime, middleware, operating system, virtualization, and physical infrastructure.

- For **SaaS**, the customer is primarily responsible for managing the data and user access. The cloud provider is responsible for securing the application, runtime, middleware, operating system, virtualization, and physical infrastructure.

4.6.5 Overview of Cloud Service Providers

Some of the major cloud service providers are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). These providers offer similar services, such as virtual machines (VMs), platform-as-a-service (PaaS), container services, and serverless functions.

The choice of a particular service could impact the region coverage, latency, and available services within a particular region. AWS and Azure both cover broader regions, while GCP has a smaller number of data centers but more advanced networking infrastructure.

The pricing differences among AWS, Azure, and GCP stem from their unique approaches to pay-as-you-go models, instance types, storage, data transfer costs, and discounts. The choice of a provider will require extensive comparison of the needed services as well as considerations for scaling.

- AWS boasts a global reach and a wide service offering, ideal for diverse applications.
- Azure excels in seamless integration with the Microsoft ecosystem and strong security features, making it a top choice for enterprises that already use other Microsoft services.
- GCP's expertise in data analytics, Kubernetes, and high-performance networking makes it attractive for data-driven and containerized applications.

4.7 SQL

4.7.1 SQL (Structured Query Language) Overview

Query Types

SQL commands are categorized into different types based on their functionality:

- **DDL (Data Definition Language)**: Used to define the structure of the database, such as creating tables, altering their structure, and defining constraints.

- **DML (Data Manipulation Language)**: Used to manipulate data in the database, including inserting, updating, deleting, and querying data.
- **DQL (Data Query Language)**: Used to retrieve data from the database using SELECT statements.
- **DCL (Data Control Language)**: Used to control access to data, such as granting or revoking permissions.
- **TCL (Transaction Control Language)**: Used to manage transactions, ensuring data integrity and consistency.

Query Execution Order

In SQL, the order of execution for a query is crucial for obtaining the desired results. The typical order of execution includes:

1. **FROM**: Specifies the tables from which data will be retrieved.
2. **JOIN**: Combines data from multiple tables based on a related column.
3. **WHERE**: Filters rows based on specified conditions.
4. **GROUP BY**: Groups rows that have the same values in specified columns.
5. **HAVING**: Filters groups based on aggregate conditions.
6. **SELECT**: Specifies the columns to be included in the result set.
7. **ORDER BY**: Sorts the result set based on specified columns.
8. **LIMIT/OFFSET**: Limits the number of rows returned or skips a specified number of rows.

Data Structures in SQL

SQL supports various data types for storing different kinds of information:

- **INT**: Represents integer values.
- **TEXT**: Stores variable-length strings.
- **VARCHAR**: Variable-length string with a specified maximum length.
- **DOUBLE PRECISION**: Represents floating-point numbers with double precision.

- **UUID**: Universally Unique Identifier for unique keys.
- **ARRAY**: Collection of elements of the same data type.
- **JSON/JSONB**: Stores JSON data in text/binary format.
- **ENUM**: Static, ordered set of values.
- **BLOB**: Stores large binary data like images or documents.

Constraints

SQL constraints ensure data integrity and enforce rules on data in tables:

- **Primary Key**: Uniquely identifies each record in a table.
- **Foreign Key**: Establishes a link between two tables based on a column.
- **Check Constraint**: Validates data based on a specified condition.
- **Unique Constraint**: Ensures that all values in a column are unique.
- **Not Null Constraint**: Ensures a column cannot have NULL values.
- **Default Constraint**: Provides a default value for a column.
- **Delete Cascade**: Automatically deletes related records in child tables when a record in the parent table is deleted.

Insert and Upsert

- **INSERT**: Adds new records to a table.
- **UPSERT**: Combines INSERT and UPDATE operations, inserting a new record if it doesn't exist or updating an existing record.

Importance of Data Integrity

Maintaining data integrity in a database ensures accuracy, consistency, and reliability of data, preventing errors and ensuring data quality.

Explain Analyze

A command used to analyze and optimize query performance in SQL, providing insights into query execution plans and performance metrics.

4.7.2 Advanced SQL Concepts

Subqueries

Subqueries are queries that are nested within other SQL statements, such as SELECT, INSERT, UPDATE, or DELETE. Subqueries can be used in various ways:

- **Correlated Subqueries**: Subqueries that reference a column from the outer query, executing the subquery for each row of the outer query.
- **Uncorrelated Subqueries**: Subqueries that can be executed independently, without referencing any columns from the outer query.

Subqueries can be used to filter data, retrieve related data, or calculate derived values. They provide a way to write complex queries in a more readable and modular fashion.

Common Table Expressions (CTEs)

Common Table Expressions (CTEs) are temporary result sets that can be referenced within a SQL statement. CTEs offer several benefits:

- Readability: CTEs can make complex queries more readable and easier to understand.
- Reusability: CTEs can be referenced multiple times within a query, reducing code duplication.
- Recursion: CTEs can be defined recursively, allowing for the creation of hierarchical or tree-like data structures.

CTEs are particularly useful when working with complex data relationships, performing data transformations, or implementing recursive queries.

Window Functions

Window functions in SQL provide a way to perform calculations across a set of rows related to the current row. Some common window functions include:

- Ranking Functions: `RANK()`, `DENSE_RANK()`, `ROW_NUMBER()`
- Aggregate Functions: `SUM()`, `AVG()`, `COUNT()`
- Offset Functions: `LAG()`, `LEAD()`
- Value Functions: `FIRST_VALUE()`, `LAST_VALUE()`

Window functions allow for the creation of complex reports and analytics without the need for subqueries or self-joins, resulting in more efficient and readable SQL code.

Conditional Statements

SQL provides several ways to handle conditional logic within queries:

- **CASE Statements**: Allow for multiple conditional checks and return different values based on the conditions.
- **COALESCE**: Returns the first non-null value from a list of expressions.
- **NULLIF**: Returns NULL if two expressions are equal, otherwise returns the first expression.

Conditional statements in SQL enable dynamic data manipulation and decision-making within queries.

Query Optimization

Optimizing SQL queries is crucial for improving performance, especially when dealing with large datasets or complex operations. Some common techniques for query optimization include:

- **Indexing**: Creating indexes on relevant columns to speed up data retrieval.
- **Query Simplification**: Removing unnecessary complexity and breaking down queries into smaller, more manageable parts.
- **Materialized Views**: Pre-computing and storing the results of complex queries for faster access.
- **Query Execution Plans**: Analyzing the plan generated by the database engine to identify and address performance bottlenecks.

By understanding and applying these advanced SQL concepts, developers can write more efficient, scalable, and maintainable database queries.

4.8 Python

4.8.1 Python Overview

Python is a versatile and powerful programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python's extensive standard library and third-party packages make it suitable for a wide range of applications, including web development, data analysis, machine learning, and automation.

4.8.2 Python Basics

- **Syntax**: Python syntax is clean and easy to understand, emphasizing readability and simplicity. Indentation is used to define code blocks, eliminating the need for braces.
- **Variables and Data Types**: Python has dynamic typing, allowing variables to change types dynamically. Common data types include integers, floats, strings, lists, tuples, dictionaries, and sets.
- **Control Structures**: Python supports common control structures like if statements, for loops, while loops, and try-except blocks for error handling.
- **Functions**: Functions in Python are defined using the `def` keyword and can take parameters and return values. They promote code reusability and modularity.

4.8.3 Text to CSV Conversion

Converting text data to CSV format is a common task in data processing. Python provides libraries such as pandas and csv to handle this conversion efficiently. These libraries offer functions to read text files, parse them, and write the data into CSV files.

Pandas Library

Pandas is a powerful library for data manipulation and analysis in Python. It provides the `read_csv()` function to read text files and the `to_csv()` function to write data to CSV files. pandas offers extensive functionality for handling tabular data, making it a popular choice for data processing tasks.[6]

NumPy Library

NumPy is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and mathematical functions, making it essential for numerical computations [5]. NumPy offers a wide range of functionalities such as array manipulation, linear algebra operations, random number generation, and Fourier transforms. Its efficient handling of large arrays and matrices makes it a cornerstone in data analysis, machine learning, and scientific research.

Matplotlib Library

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python [4]. It provides a wide variety of plotting functions and tools for producing high-quality figures in a variety of formats, including PNG, PDF, SVG, and many others. Matplotlib integrates well with other data analysis libraries like NumPy and Pandas, allowing seamless visualization of numerical data. Its versatility and extensive customization options make it a powerful tool for data visualization in various domains, such as scientific research, finance, and business analytics.

Pillow Library

Pillow, also known as the Python Imaging Library (PIL), is a powerful image processing library for Python [7]. It provides extensive support for opening, manipulating, and saving different image file formats, including PNG, JPEG, GIF, and TIFF. Pillow offers a wide range of image processing capabilities, such as image resizing, filtering, color manipulation, and drawing operations. Its comprehensive functionality and ease of use make it a valuable tool for tasks like image analysis, computer vision, and image-based applications in various fields, including web development, graphics processing, and multimedia.

Csv Module

The csv module is part of Python's standard library and provides functions for working with CSV files. It includes functions like `csv.reader()` and `csv.writer()` to read and write CSV files, respectively. While not as feature-rich as pandas, the csv module is lightweight and suitable for simple text to CSV conversion tasks.

4.8.4 Image Processing

Image processing in Python involves manipulating and analyzing images using libraries such as OpenCV and Pillow. Common tasks include resizing, cropping, filtering, enhancing, and detecting objects in images.

4.8.5 Sound Processing

Sound processing in Python is facilitated by libraries like Librosa and SciPy. It includes tasks such as audio file manipulation, feature extraction, noise reduction, and signal processing techniques like Fourier transforms.

4.8.6 Progress Bar with tqdm

The tqdm library provides a simple and elegant way to add progress bars to Python code. It enhances user experience by visualizing the progress of loops and operations, especially in tasks with lengthy computations or iterations.

4.8.7 Generating Fake Data with Faker

Faker is a Python library used to generate fake data for testing and development purposes. It offers a wide range of data types, including names, addresses, phone numbers, dates, and lorem ipsum text, helping developers create realistic test scenarios.

Librosa

Librosa is a Python library for audio and music analysis. It provides tools for feature extraction, spectrogram generation, beat tracking, tempo estimation, and other tasks related to audio processing and analysis.[3]

SciPy

SciPy is a Python library for scientific and technical computing. It builds on top of NumPy and includes modules for optimization, integration, interpolation, linear algebra, signal processing, statistics, and more, making it a powerful tool for various scientific applications.[8]

4.8.8 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a classic problem in computer science and optimization. Python offers various algorithms and libraries, such as NetworkX and SciPy, to tackle TSP and find optimal or approximate solutions, making it a popular choice for solving

combinatorial optimization problems.

4.8.9 Assignments

Assignment 1

After the learning of html and css, I was given my first task as project building to convert the Figma template to static web page.

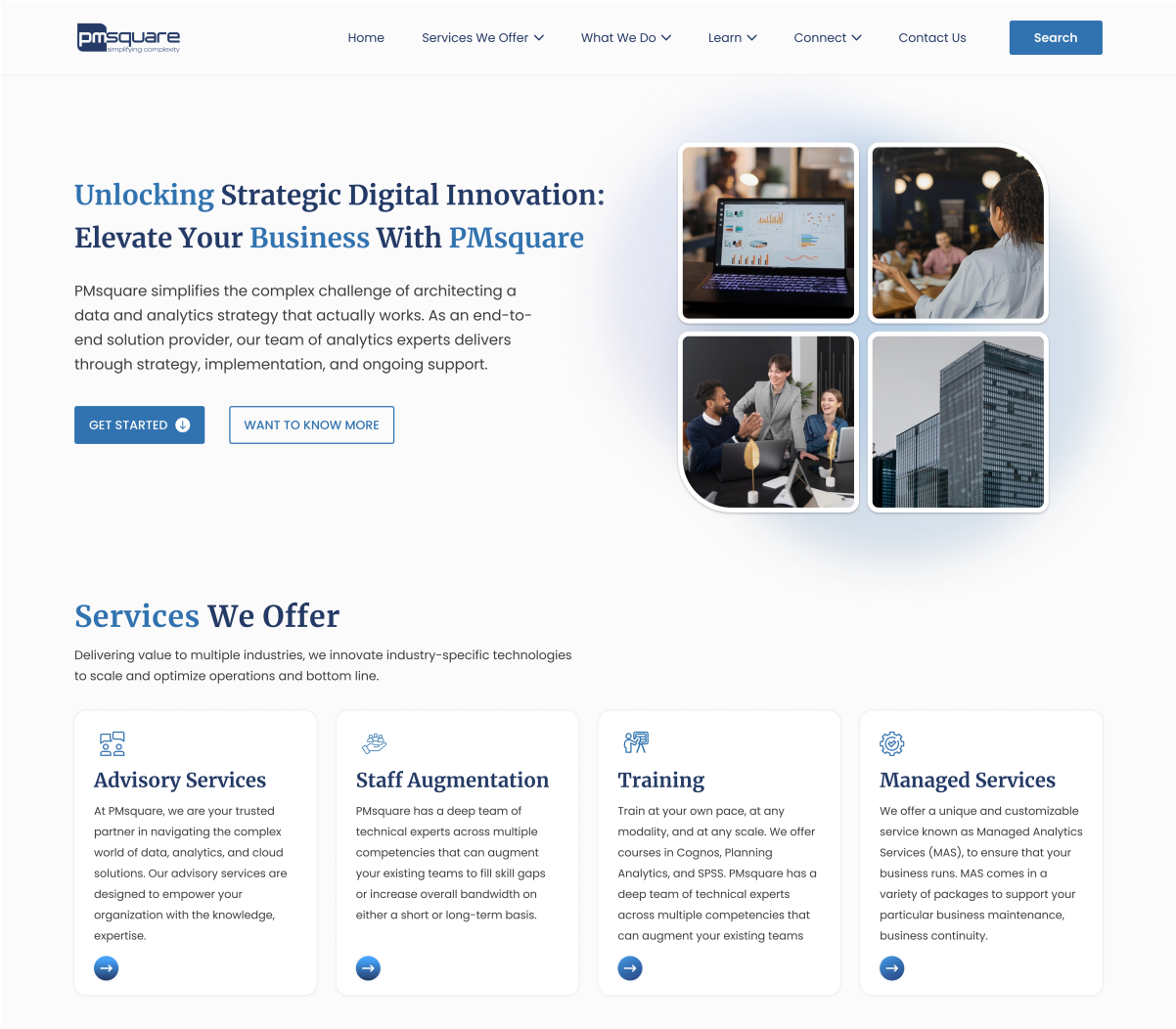

Figure 4.1: Finalized PMsquare Design

Inside it, my learnings were:

1. Challenges with Flexbox and Gridbox:

• While working on the PMSquare template, I encountered some difficulties distinguishing between the application of Flexbox and Gridbox for layout design. Particularly, I found it chal-

lenging to determine when to utilize Flexbox for flexible layouts and when to opt for Gridbox for grid-based designs, especially in defining the structure for sections like the navbar, hero, and footer. Through persistent exploration and research, I gradually gained clarity on their respective functionalities and applications, enabling me to effectively structure the layout.

2. Issues with Image Downloads from Figma:

• During the design phase, I faced hurdles in downloading images from Figma for integration into the PMSquare template. Initially, I encountered compatibility issues and compromised resolution when using traditional image formats. However, after exploring alternative solutions, I discovered the advantages of downloading images in SVG format. Embracing this best practice, I successfully overcame the challenges associated with image downloads, ensuring seamless integration of high-quality visuals into the template design.

3. Maintaining Professionalism in Section Naming:

• Ensuring adherence to professional and formal standards, I meticulously assigned clear and descriptive labels to the sections within the project template, maintaining consistency and clarity in the naming convention throughout the design process. It was essential to ensure that section names reflected a professional tone and conveyed the intended purpose clearly. By adhering to these standards, I aimed to enhance the template's overall professional appearance and user experience.

4. Segmenting Hero Section with Image Containers:

• Within the hero section of the PMSquare template, I implemented a segmentation approach by dividing it into two distinct sections. On the right side, I incorporated an image with four containers, each containing individual images arranged as the nth child. Additionally, I observed a visual effect resembling a box shadow in the background, which upon closer inspection, turned out to be another image. Recognizing this, I integrated it as a separate image element, maintaining visual coherence and aesthetic integrity within the hero section.

Third learning task was assigned to me for Tailwind CSS. My learning was Tailwind CSS offers a unique way to style web apps by using utility classes directly on HTML elements, skipping the need for custom CSS. This speeds up development because developers can quickly apply styles without writing extra CSS code. It also encourages creating reusable components with clear HTML markup, making the code easier to maintain and collaborate on.

Assignment 2

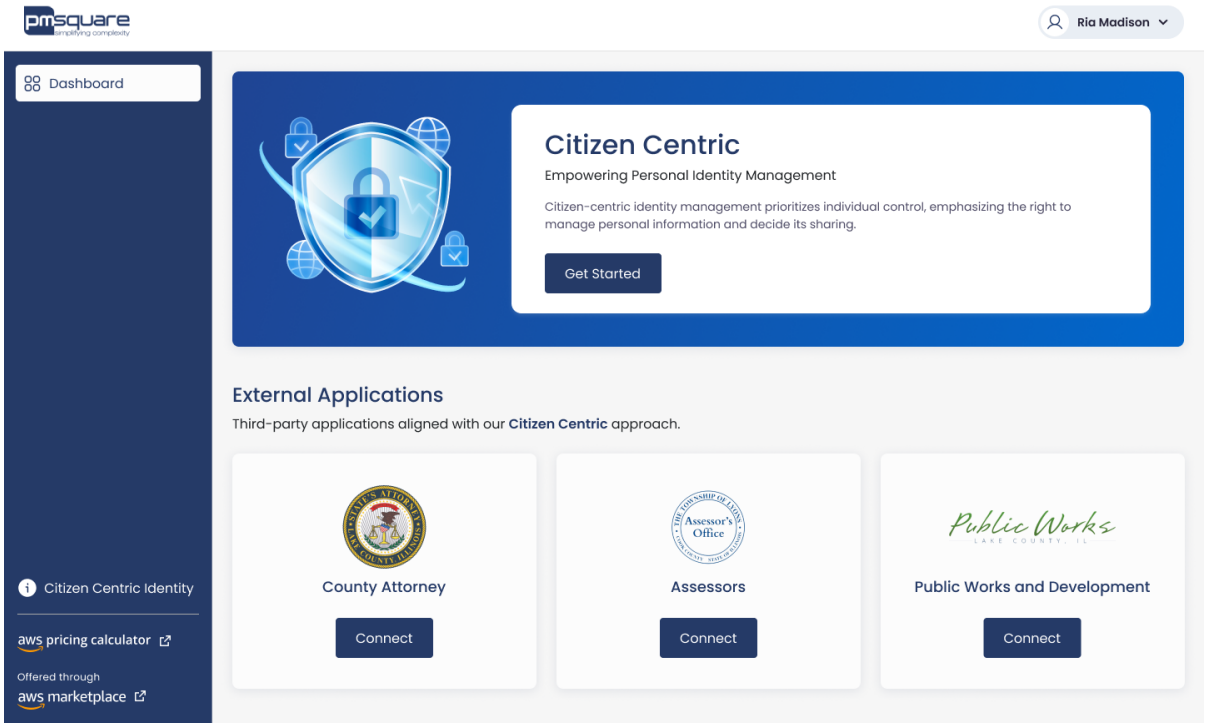

Figure 4.2: Landing Page

In my pursuit of advancing user-centric design principles, I undertook a project focused on creating a citizen-centric website using Tailwind CSS. With a strong commitment to enhancing user experiences, I set out to explore Tailwind CSS's utility-first approach to design, aiming to optimize accessibility and usability for all citizens accessing the platform.

Throughout the project, I immersed myself in Tailwind CSS's methodology, harnessing its utility classes to prioritize user needs and preferences at every stage of development. By skillfully employing flexbox and grid utilities, I crafted intuitive layouts that prioritized clarity and ease of navigation, ensuring citizens could seamlessly interact with the website across devices and screen sizes. Additionally, I integrated Tailwind CSS seamlessly with

npm and npx, streamlining the development process and enabling rapid iteration to address evolving citizen requirements.

This project has deepened my understanding of user-centric design principles and equipped me with valuable insights into leveraging Tailwind CSS to create citizen-centric digital experiences. With a strong foundation in Tailwind CSS's utility-first approach and best practices, I am poised to continue advocating for user-centric design and delivering innovative solutions

that prioritize the needs and preferences of citizens. Armed with Tailwind CSS, I am empowered to build accessible, inclusive, and intuitive digital platforms that foster positive citizen engagement and enhance overall satisfaction.

5. Internship Plan

The internship plan is structured to cover a broad range of topics and technologies relevant to modern software development. It aims to provide exposure to various areas including:

- Foundational web technologies: HTML, CSS, JavaScript
- Advanced web development concepts: React, TypeScript
- Cloud technologies
- Python programming: including image and audio processing
- Database management and querying (SQL)
- Prototype app development

The plan is designed to start with foundational concepts and gradually progress to more advanced topics, allowing the intern to build a solid understanding of different areas in software development.

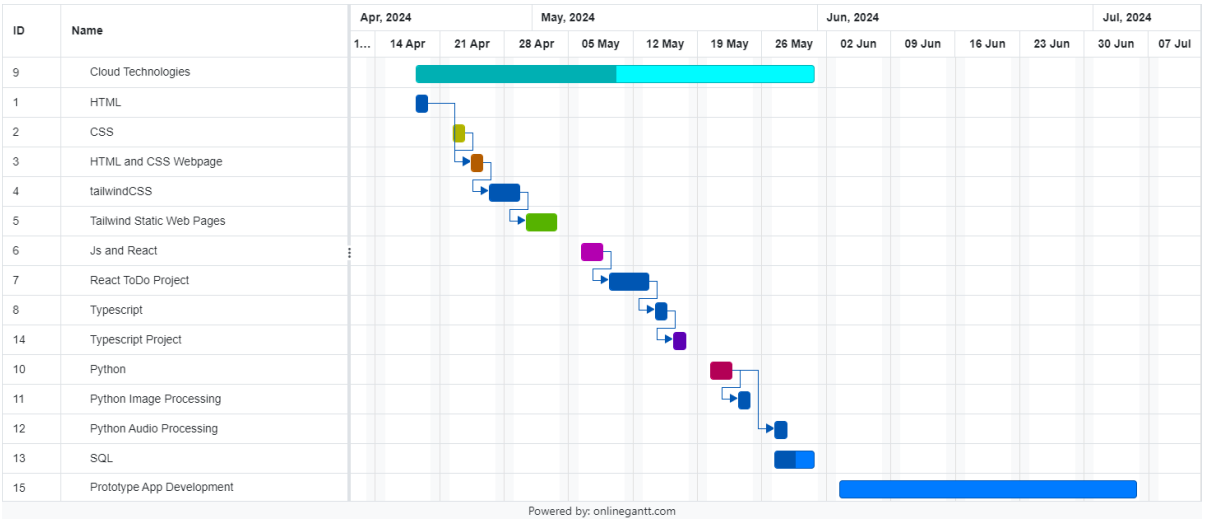

Figure 5.1: Internship Gantt Chart

6. Expected Outcome of Internship Activities

The internship activities are expected to yield the following outcomes:

- A strong foundation in various technologies relevant to software development.
- Practical experience in web development, cloud technologies, Python programming, and database management.
- Exposure to industry-relevant tools and technologies.
- Ability to apply acquired knowledge and skills to real-world scenarios, demonstrated through the prototype app development task.
- Enhanced problem-solving and critical thinking skills through hands-on experience.
- Preparedness for a career in software development or related fields.

Overall, the internship aims to equip the intern with a diverse skill set and practical experience, making them a valuable asset for potential employers in the software development industry.

References

[1] Amazon Web Services. Aws documentation. [Online], 2024. Available at https://docs.aws.amazon.com/.

[2] Amazon Web Services. Aws services. [Online], 2024. Available at https://aws.amazon.com/products/.

[3] The Librosa Development Team. Librosa library. [Online], 2024. Available at https://librosa.org/doc/latest/.

[4] The Matplotlib Development Team. Matplotlib library. [Online], 2024. Available at https://matplotlib.org/stable/contents.html.

[5] The NumPy Development Team. Numpy library. [Online], 2024. Available at https://numpy.org/doc/stable/.

[6] The Pandas Development Team. Pandas library. [Online], 2020. Available at https://pandas.pydata.org/docs/user_guide/index.html.

[7] The Pillow Development Team. Pillow library. [Online], 2024. Available at https://pillow.readthedocs.io/en/stable/.

[8] The SciPy Development Team. Scipy library. [Online], 2024. Available at https://docs.scipy.org/doc/scipy/reference/.