# A TECHNICAL REPORT ON STUDENTS INDUSTRIAL WORK EXPERIENCE SCHEME(SIWES)

**UNDERTAKEN AT**

**4, BALARABE MUSA CRESCENT, VICTORIA ISLAND, LAGOS, NIGERIA.**

BY

**IDOGUN JOHN OWOLABI**
**CPE/15/2418**

**SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING**

January 13, 2020

DEPARTMENT OF COMPUTER ENGINEERING
FEDERAL UNIVERSITY OF TECHNOLOGY, AKURE

# Certification

This is to certify that this report is a detailed account of Students' Work Experience Scheme (SIWES) undertaken by Mr. JOHN OWOLABI IDOGUN with Matriculation Number CPE/15/2418 for a period of six months at *ip*NX Nigeria Limited, Victoria Island, Lagos State and has been compiled in accordance to regulations guiding the preparation of reports in the department of Computer Engineering, Federal University of Technology, Akure, Ondo State.

______________________________ ______________________________

SIWES Coordinator's Signature Head of Department's Signature

______________________________

Student's Signature

# Dedication

*To my wonderful Guardians, benefactors and sponsors, Engr. & Mrs. Sunday Idogun: know that I unimaginably cherish the lessons you taught me which make me belief that with God, smart work, patience, and determination, anything can be achieved.*

*To all tech-savvies: we have got a lot to learn.*

**Idogun John O.**

# Acknowledgements

I owe a huge debt of gratitude to God Almighty for His unconditional and resilient love cum compassion towards me. He is deeply appreciated for the intellectual acumen He amply bestowed on me before, during and after this industrial training.

This project-based internship was made possible by funding from my Guardians, benefactors and sponsors, Engr. & Mrs. Sunday Idogun, my amiable sister, Mrs. Modupe Igbekoyi, and her husband, Mr. Idowu Igbekoyi, my great brother, Mr. Festus Idogun, and all my family members. You have all been awesome to me.

I would like to thank my industry-based supervisor, Oluboyo Charles Oluwaseun, for his encouragement, supports in all aspects, and timely help. I am favoured to have been under your tutelage. Special thanks are extended to Frances Nnamadim, a Business Analyst and UI/UX designer, who never seemed to be tired of imparting knowledge and writing comments as well as remarks, for discussions, feedback, ideas, and for help on the various projects done. I would also like to thank Amuche Benson-Onyeibor, who extended her generosity and was kind enough to offer comments and suggestions in the course of implementing some projects. My grateful thanks go to all the people at *ip*NX Nigeria Limited for the pleasant co-operation and hospitality they offered during my stay, particularly Shade Efiong-Bassey, the Human Resources Managers, and Administrators.

Finally, I want to thank Dr. (Mrs) Folasade Mojisola Dahunsi, Ag. Head Of Computer Engineering Department, Federal University of Technology, Akure, especially for help on getting Internship placement and superb cum expressive motherlike love.

# Abstract

This paper reports a six-month Internship with the Research and System Architecture department at *ip*NX Nigeria Limited, Victoria Island, Lagos.

My tasks were to engineer, develop and deploy customer-centric software products while employing modern Software development paradigms and practices. Some projects require data visualization in which communication between Front-end and Back-end asynchronously JavaScript Object Notation (JSON) over eXtensible Markup Language and HyperText Transfer Protocol Request (XHR) using Asynchronous JavaScript and eXtensible Markup Language (XML) (AJAX) while others involve implementing complex algorithms such as the Materialized Path Algorithm to provide some required complex features as contained in the Software Requirement Specification document. Such implemented complex features include but not limited to a Threaded Commenting System, Full-text search system using PostgreSQL's full-text search engine, a live chat system and other basic Create, Retrieve, Update and Delete (CRUD) processes using Flask and Django frameworks as well as PostgreSQL and SQLite databases. Most of the software projects were wholly written in Python, an interpreted, high-level, general-purpose programming language created by Guido van Rossum and first released in 1991 that lets one work quickly and integrate systems more effectively. Few other projects such as a Web crawler and Recommender system were implemented in Julia, a new programming language offering a unique combination of performance and productivity that promises to change scientific computing, and programming in general. Julia picks the best parts of existing programming languages, providing out-of-the-box features such as a powerful Read-Execute-Print Loop (REPL), an expressive syntax, Lisp-style metaprogramming capabilities, powerful numeric and scientific programming libraries, a built-in package manager, efficient Unicode support, and easily called C and Python functions.

System/Linux Administration works were also embarked upon ranging from deploying already developed applications to a Linux server using Apache2 and nginx to complying with versioning and back-end compatibility of resources. In cooperation with a notable senior staff in the department of Business Intelligence and Data Analytics (BIDA), I designed and developed from scratch a complex blogging web application primarily to bring together tech enthusiasts. Syntax highlighting was implemented using Primejs and Django-ckeditor was used as ”What You See Is What You Get (WYSIWYG)” rich texts or documents editor.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **AJAX** | Asynchronous JavaScript and XML |
| **API** | Application Programming Interface |
| **CRUD** | Create, Retrieve, Update and Delete |
| **DTL** | Django template language |
| **HTML** | HyperText Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **JSON** | JavaScript Object Notation |
| **ORM** | Object Relational Mapper |
| **REPL** | Read-Execute-Print Loop |
| **SDLC** | Software Development Life Cycle |
| **SIWES** | Students' Work Experience Scheme |
| **SQL** | Structured Query Language |
| **vCPE** | Virtual Customer Premises Equipment |
| **XHR** | eXtensible Markup Language and HyperText Transfer Protocol Request |
| **XML** | eXtensible Markup Language |
| **XHTML** | eXtensible HyperText Markup Language |
| **XSLT** | eXtensible Stylesheet Language Transformations |
| **WSGI** | Web Server Gateway Interface |
| **WYSIWYG** | What You See Is What You Get |

# List of Symbols

| | |
|---|---|
| $\alpha$, $\beta$ | Damping constants |
| $\theta$ | Angle of twist, rad |
| $\omega$ | Angular velocity, rad/s |
| $b$ | Width of the beam, m |
| $h$ | Height of the beam, m |
| $\{f(t)\}$ | force vector |
| $[K^e]$ | Element stiffness matrix |
| $[M^e]$ | Element mass matrix |
| $\{q(t)\}$ | Displacement vector |
| $\{\dot{q}(t)\}$ | Velocity vector |
| $\{\ddot{q}(t)\}$ | Acceleration vector |

# Chapter 1

# Introduction to SIWES Program

## 1.1 Background

The SIWES is an undergraduate training programme with specific learning and career objectives geared towards the development of occupational and industrial competencies of the participating students. It is a requirement for the award of degrees and diploma to all students of tertiary institutions in Nigeria pursuing courses in specialized engineering, technical, business, applied sciences and applied arts. The scheme is a three-party program which involves the student, the tertiary institution, and the industry,and it exposes students to practical knowledge of their course of study. Furthermore, SIWES is a general programme cutting across over 60 programmes in the universities, over 40 programmes in the polytechnics and about 10 programmes in the colleges of education. Thus, SIWES is not specific to any one course of study or discipline. Consequently, the effectiveness of SIWES cannot be looked at in isolation with respect to a single discipline, hence it is better explored in a holistic manner since many of the attributes, positive outcomes and challenges associated with SIWES are common to all disciplines participating in the scheme. Hence, the approach of this report is to look at SIWES as a general study programme cutting across several disciplines. Furthermore, the report gives details of the industrial work experience gained at *ip*NX Nigeria Limited, Victoria Island, Lagos state.

## 1.2 Brief History of SIWES Program

The SIWES started with about 748 students from 11 institutions of higher learning in 1974. By 1978, the scope of participation in the scheme had increased to about 5,000 students from 32 institutions. The Industrial Training Fund, however, withdrew from the management of the scheme in 1979 owing to problems of organizational logistics and the increased financial burden associated with the rapid expansion of SIWES. Consequently, the Federal Government funded the scheme through the National Universities Commission (NUC) and the National Board for Technical Education (NBTE) who managed SIWES for five years (1979 – 1984). The supervising agencies (NUC and NBTE) operated the scheme in conjunction with their respective institutions during this period

## 1.3 Scope of SIWES

The decree which legislates this scheme does not give a time confinement or restriction to its implementation but it is a relative assignment. This implies that each school has the mandate with respect to her academic calendar to categorically state when the scheme will be undertaken but must ensure that the required period of training is satisfied thus.

## 1.4 Aims and Objectives of SIWES

The objectives of SIWES as stated in Information and Guideline for SIWES (2002) are:

1. To provide an avenue for students in higher institutions to acquire industrial skills and experience in their approved course of study.
2. To prepare students for the industrial works situation which they are likely to meet after graduation.
3. To expose students to work methods and techniques in handling equipment and machinery not available in their institutions.
4. To provide students with an opportunity to apply their knowledge in real work situation thereby bridging the gap between theory and practical.
5. To enlist and strengthen employers' involvement in the entire educational process and prepare students for employment in Industry and Commerce.

# Chapter 2

# *ip*NX Nigeria Limited

## 2.1 History and Corporate Profile

### 2.1.1 History

With its headquarters at 4, Balarabe Musa Crescent, Victoria Island, Lagos and several branch offices in Lagos, Ibadan, Abuja, Kano and Port Harcourt, *ip*NX Nigeria Limited is a leading provider of infrastructure-based telecommunication and information technological services based here in Nigeria. With more than a decade of experience, the company was formed by the divestment of the telecommunications services division of Telnet Nigeria Limited and has been in operations for over fifteen (15) years. *ip*NX Nigeria Limited started as a business division of Telnet Nigeria Limited - the leading indigenous Telecommunication and Information Technology Services Company in Nigeria. Telnet started business in 1987 as telecommunications engineering company and grew into other areas of information technology as technology evolved and opportunities arose. A major part of the business of telnet was providing data communication services, mainly wide area networks to Corporate communities in Nigeria in the Oil and Gas as well as Financial Services industries. Since NITEL was the monopoly provider of telecommunication services in Nigeria due to government regulation, these networks had to be built with NITEL facilities. In December 1992, the Federal Government of Nigeria deregulated the telecommunications industry, thereby opening it up to competition and other organizations were allowed to provide telecommunications services. Telnet saw this as an opportunity to improve its services to its customers and started to build its own communications network using radio and VSAT (satellite) technologies. The radio networks are utilized for communications between locations in the same metropolitan area while VSAT networks were mainly used to provide long distance communications. The corporate organizations used the networks provided for both private voice and data communications.

In 2001 Telnet decided to separate the infrastructure based business from the engineering (or knowledge) based business. This was done to:

- To allow Telnet Nigeria Limited provide engineering services to other infrastructure based service providers who might see Telnet as a competition.
- To allow other investors to invest in the infrastructure based business, which is very capital intensive.

The infrastructure based Services Company within Telnet was therefore detached from the group to form a new company – "Netco Services Limited". Some of the Nigerian Communications Commission (NCC) licenses with Telnet were transferred to Netco and Netco obtained some additional licenses from the Nigerian Communications Commission (NCC). In total Netco has:

- Regional 3.5GHz FWA (Fixed Wireless Access) licenses in Lagos, Cross River, Bayelsa and Abuja.
- National VSAT license.
- Internet Service Provider License.

Unfortunately, a subsidiary of the Nigerian National Petroleum Company (NNPC); the National Engineering and Technical Company had also been known in the Nigerian environment with the acronym Netco and this caused a bit of confusion in the market place, hence in April 2003 the name of the company was changed to *ip*NX Nigeria Limited.

*ip*NX has now obtained additional frequencies from the NCC to be used to provide consumer and small businesses with Internet and Data Communications. These have been allocated to Lagos, Abuja (FCT), Cross Rivers and Bayelsa States with an opportunity to go to other states in Nigeria after we have started providing services in these locations.

### 2.1.2 Corporate Profile

*ip*NX is one of Nigeria's fastest growing Information and Communications Technology companies, serving a multitude of needs across enterprises, small businesses and residents with innovative, world-class services.

Our ability to identify, satisfy and exceed today's market needs is a testament to over a decade of experience, our commitment, drive and passion realized through highly skilled and well seasoned professionals.

As a pioneer and a leading Fibre-To-The-Home (FTTH) operator in Nigeria, we currently provide a number of solutions to various industries and market segments using industry-leading technology (such as our very own Fibre-To-The-Home (FTTH) cable technology) as our core access network infrastructure and fixed wireless radio services (via licensed frequency).

We also proffer complementary IT solutions, with a view of covering key commercial and suburban regions.

### Vision Statements of *ip*NX

To be the preferred communications and IT enabler in Africa and beyond.

### Mission Statements of *ip*NX

Leveraging technology to create innovative solutions that help mankind thrive.

### Core Credentials

Over 15 years experience

Trustworthy and devoted

More than 5,000 large and small business customers

Over 800km of cutting-edge fibre-optic cable infrastructure

*ip*NX is also the dependable service provider to the financial services sector, connecting all banks in Nigeria, Central Bank, InterSwitch & NIBSS.

### Services

Broadband Internet

IP-VPN/MPLS

IP Wholesale

Telephony

Collocation & Hosting

Unified Communication

Cloud Computing

## 2.2 Why *ip*NX?

At *ip*NX, we pride ourselves on being game changers. We value smart, talented, hard-working people who stand tall and resolute amidst challenges. If great ideas thrill you, then you will fit in with us at *ip*NX. You will enjoy a stimulating environment, totally cool colleagues and all the opportunity your drive can handle.

### 2.2.1 Our Culture

Everyone in *ip*NX has the mindset of the explorer. We are always bold. We are to known venture into unknown and uncharted territories, yielding technology - enabled communications solutions that deliver optimal value to our stakeholders.

We constantly innovate and break boundaries. We are relentlessly customer obsessed, starting all we do with our customer in mind.

At *ip*NX, each employee takes personal responsibility and full accountability for up-holding a culture that is inclusive, ethical and supportive of our corporate vision, goals and value.

### 2.2.2 Our Guiding Principles

We don't go out of our way to be different - We are. Our exciting and unique office culture is driven by who we are in the real world. From the word go, we focus on providing every person, every home and every business in Nigeria with world-class information, communication and entertainment services, while having fun.

#### Customer Obsession

At *ip*NX we obsess about our customers, and how to meet their requests. We pay attention to every detail that concerns them, with patience, true care, concern, constant follow up and follow through. We start and end all we do with our customers in mind.

#### Explorer

We are willing to try something new, venture into unchartered territories. Have a Sense of adventure, dare to be different. Be bold!

#### Can Do Spirit

Anything is possible! With hard work and strong commitment there is nothing we cannot achieve.

#### Teamwork

By working together, complementing each other's diversity, we achieve more.

#### Innovation

We constantly innovate in our processes, technology, products and services to deliver value.

#### Continuous Improvement

We continuously look to improve everything we do.

#### Bias for Action

Speed matters in business. We push relentlessly for desired outcomes and focus on results… the process is not the output! We value calculated risk taking.

#### Excellence

We are committed to relentlessly pursue excellence.

#### Continuous Learning

We will continuously improve our skills and knowledge in order to remain the best at what we do.

#### Respect for the Individual

Creating an open and enabling environment where different opinions can be expressed without fear. We value the contribution of everybody at every level. We treat our colleagues, whether supervisors, peers, or subordinates, with courtesy and respect, without harassment, or physical or verbal abuse. We acknowledge our rights to diversity yet are united in achieving our corporate goals.

#### Professionalism

At all times, we will exhibit strict compliance to the tenets of our profession and work environment.

**Integrity**

We deliver on promises to ourselves, clients and colleagues, holding ourselves to high levels of ethical behavior.

**Ownership**

Ownership is about taking responsibility and accepting accountability. I am accountable for the timeliness and quality of an outcome designated to me, even when working with others.

**Frugality**

We will achieve more with less, without compromising on quality. Our frugality inspires us to be resourceful, innovative and self-sufficient.

**Exciting Work Environment**

We believe our workplace should also be fun and creative, making it a place where people want to be every minute; where ideas thrive. This makes us more productive and keeps us engaged. We work smart …and we play smart.

### 2.2.3 Training And Development

We recognize at *ip*NX that learning improves and drives the company to its success. Our commitment to learning and development at *ip*NX is not just lip service but an actual dedicated effort that strengthens and empowers our employees throughout the duration of their career with us.

We offer amazing internal resources that include:

- ICademy online
- Employee development programmes
- Mentorship programmes
- Leadership development programmes

### 2.2.4 Compensation and Benefits

Work can be demanding but fun. However, at *ip*NX, we ensure that you stay rewarded for your performance and contribution. Compensation is determined by a number of factors including company performance, divisional performance, and individual performance.

Our benefits generally include:

- Insurance
- Paid vacation time
- Paid paternity and maternity leave
- Other position specific benefits, amidst other options

### 2.2.5 Corporate Social Responsibility

Our aim is to support our communities by empowering them primarily through technology-enabled. Leveraging on technology, we contribute to the growth of the young population preparing them for independence and helping them to become useful members of society while achieving self-actualisation.

Through the use of technology, we provide educational services that close the gap between our children and children in a more developed economy.

Over the years, *ip*NX has improved the quality of education in the communities where she has done business.

## 2.3 *ip*NX Organogram

*ip*NX, being a big company, is divided into various divisions with divisional Chief Executive Officers (CEOs) all reporting to the Group Managing Director (GMD), Ejovi Aror. From the foregoing, *ip*NX currently has the organogram shown in Figure 2.1.

## 2.4 *ip*NX Departments And Divisions

*ip*NX provides a range of network connectivity, and delivery of internet, telephony, television as well as cloud-based software application services for our clients. These services are delivered by teams working across multiple divisions and departments.

### 2.4.1 *ip*NX Departments

#### Human Capital Management

Our Human Resource team brings out the best potential in every staff because here *ip*NX, our people matter. Their talent, commitment, energy – plus the million and one

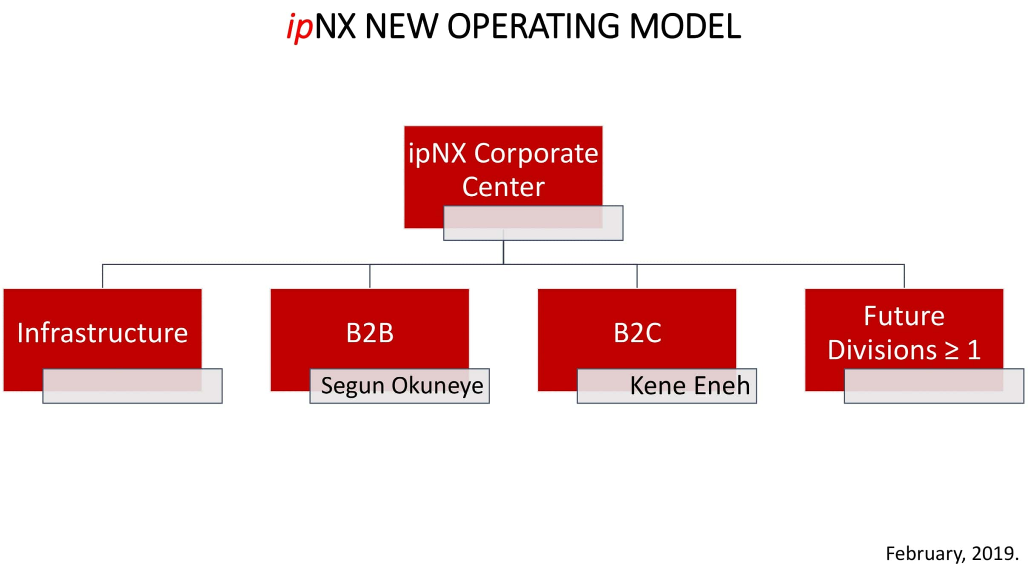
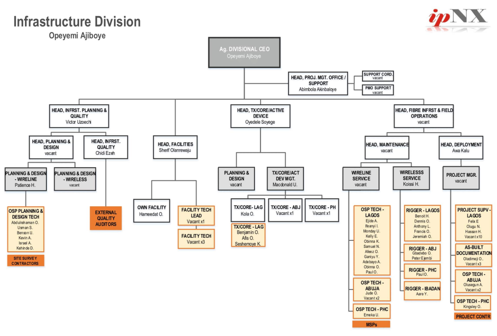
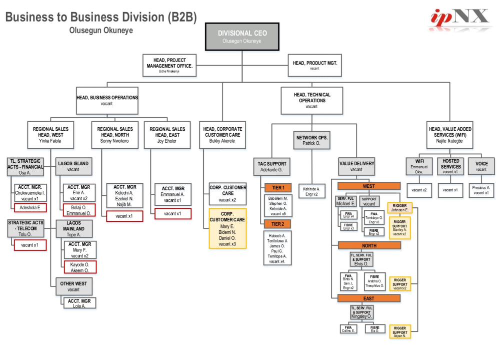


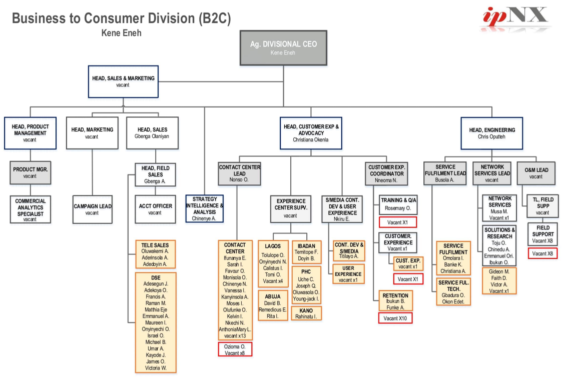
Figure 2.1: *ip*NX Main Organogram



(a) Infrastructure Organogram



(b) Business Division(B2B) Organogram



(c) Retail Division(B2C) Organogram

Figure 2.2: Each division's organogram

things they bring to our business- makes us who we are. The way you think matters to us. we value creative thinking. So far, it has helped us recreate rules and we want more of it. If you like adventure, if you don't give up, if you are focus on getting results, we want you at *ip*NX.

### Finance

We would be nowhere without the expertise and insight of our finance teams. Our various teams include: Financial Accounting Revenue Assurance Treasury Billing The teams are there to understand, support and challenge the business, making sure we are doing everything we can to run efficiently by providing the financial expertise required to grow our business to profitable levels and heights. If you have an eye for numbers and a brain for planning, what are you waiting for?

### Legal, Risk And Compliance

Our Legal area has a big impact on our actions. Fortunately, we can rely on the counsel, creative thinking and sound judgment of the Law, Risk and Compliance team to unravel the finer points and keep us on the right track. If you know you are a champion of business integrity; are good at building constructive business relationships with our customers, partners, local communities, or the government; strive to protect and promote innovation, enterprise, then join us for a career with our LRC team.

### Customer Experience & Advocacy

When our customers need support and advice, or they want to get something sorted out, they deserve the right help, right on time. Everyone in customer service loves to make a difference because they strive for excellence. They won't stop until they can see the smile of satisfaction on the customer's face. Very simply put, we obsess over our customer. On joining the Customer service team, you will be the voice of *ip*NX. And whatever the next call brings, your goal remains the same; to delight and dazzle with your knowledge and commitment. If you can throw in some pleasant surprises along the way, everyone is a winner.

### Health, Safety And Environment

*ip*NX Health, Safety and Environmental team is responsible for development, oversight, and management of environmental health and safety programs that protect the *ip*NX working environment, provide safe and healthy conditions for work and comply with applicable laws and regulations. It is the team's goal is to pro vide responsive service and critical support to ensure that *ip*NX is a safe and healthy environment in which to work, study, and live. If you are an expert in what you do and love safety, apply to join our HSE team.

**Marketing**

*ip*NX Marketing Department plays a massive role in promoting the mission and business of *ip*NX. *ip*NX Marketing team's main focus is to understand our customers, their values, and perception and make sure they get it – through our amazing services because *ip*NX was born to disrupt – but with a purpose. If you can wow our customers with *ip*NX brand, you might just fit right in with this team.

**Internal Audit**

Our Internal Audit team provides *ip*NX with unbiased, objective and constructive views that help companies succeed. Our internal audit team examines and evaluates the policies, procedures and systems which are in place to ensure the reliability and integrity of information, compliance with policies, plans, laws and regulations; the safeguarding of the assets; and the efficient use of resources. We are looking for professionals who take pride in their work.

**Corporate Services**

Our Corporate services are made up of Human Capital Management, Administrative Services, Facility Management, etc. They provide services to our people. They make sure we get the best employees who remain the best employees on top of their game; they make sure we have both the physically and mentally enabling environment to support and push out our creativity. At *ip*NX, we want to work with people who know the next best idea is just around the corner – and who have the ambition, creativity, and passion for going after it.

**Supply Chain Management**

Our SCM team helps cordinate and integrate seamless information, material and finance flow from suppliers to customers, so that we have an excellent customer service, an ideal inventory management and low unit cost. At *ip*NX, our SCM team has the prime responsibility of integrated planning, sourcing and deployment of material connected to our network services . If you know you have the expertise for this job, why not join our SCM team.

#### Information Systems And Technology

As an organization that is in love with leading edge innovation and services, it is no wonder that our IS&T team plays a crucial role. The team provides exceptional technological services and support. Our team enables efficiency, productivity and simplicity both internally and externally, helping us to grow. Joining our IS&T team means you will help keep our internal IT running smoothly by creating and maintaining our systems and applications.

#### Fibre Project And Rollout

Our Fibre Project and Rollout team are our hunchos when it comes to rolling out our network fibres direct to business areas and homes. If you have a passion for success and want to make a difference for customers every day, join our Fibre Project and Rollout Team.

#### Admin Services

Every organization needs somebody to coordinate it. That job is for the *ip*NX administrative team. The administrative team brings its conscientious planning, attention to detail and flexible thinking to the heap of projects. They help make sure things run like a well-oiled clock. This means organizing everything from staying on top of day to day administration, prioritizing and managing real estate and other assets, getting involved in team strategy and company – wide objectives. If you want to join our administration team, there is always plenty of opportunity to make your mark.

#### Research And System Architecture

The *ip*NX Research and System Architecture team is reinventing the technology world every day. The Research and System Architecture team is in charge of all our research and development and are also responsible for coming up with tomorrow services that meet the needs for our today's customers. They look for ways to do it cheaper, faster, and better. We are looking for those who have the drive and passion for technology.

### 2.4.2 *ip*NX Divisions

#### Infrastructure Division

*ip*NX network is always on duty, always in demand. That means it has to be reliable. This all comes down to the diligence and support of our network services team. Whether fixed-voice and data; mobility; and IT services, they make sure our network is ever available. This means our customers' experience is in their hands. They are there to trouble shoot malfunctions and make the network service bigger, better, faster and stronger. If you are someone who can resolve issues, develop ideas to enhance a growing network and most importantly, work with the *ip*NX team and embrace the values we uphold, then you have what it takes to be part of *ip*NX community.

#### Retail Division

#### Business Division

# Chapter 3

# Projects Done and Experiences Gained

## 3.1 Preliminaries

This section gives a brief introduction to or exposes the technologies used during in the course of implementing the projects assigned to me.

### 3.1.1 Software development process overview at *ip*NX

In IEEE standard Glossary of Software Engineering Terminology, the SDLC is: "The period of time that starts when a software product is conceived and ends when the product is no longer available for use". This circle typically includes the following stages:

1. Planning and requirement analysis: Each SDLC model starts with Planning and requirement analysis, in which stakeholders of the process plan and discuss the requirements for the final product with the goal of a detailed definition of the system requirements.

2. Designing project architecture: In this stage, the developers engage in designing the architecture of the software product and any surfaced technical questions are discussed by all the stakeholders, including the customer. Technologies which will be used, team load, limitations, duration and budget are also defined.

3. Development and programming: Having approved the requirements stated, the process comes to this stage – actual development. Programmers incept writing of source codes to align with the defined requirements, System administrators adjust the software environment, front-end programmers develop the user interface of the program and the logics for its interaction with the server. The programming by itself assumes four stages:

- Development of Algorithms
- Writing of Source Codes
- Source codes compilation
- Unit testing and debugging

4. Testing This is the phase where comprehensive testing and debugging of the developed software product take place. All the code flaws missed during the development are detected, documented, and passed back to the developers to fix. The testing process repeats until all the critical issues are removed and software workflow is stable.
5. Deployment: When the program is finalized and has no critical issues, it is time to launch it for the end users and this is what happens in this stage.
6. Maintenance

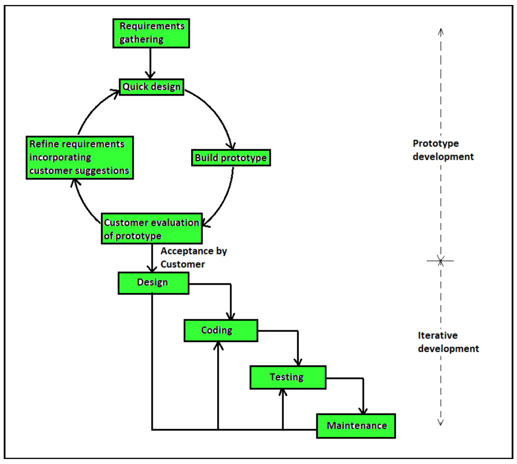
According to Sharma and Singh (2015), General software process models are:

1. Waterfall model
2. Prototype model
3. Rapid application development model (RAD)
4. Incremental model
5. Spiral model
6. Agile
7. V-shaped model

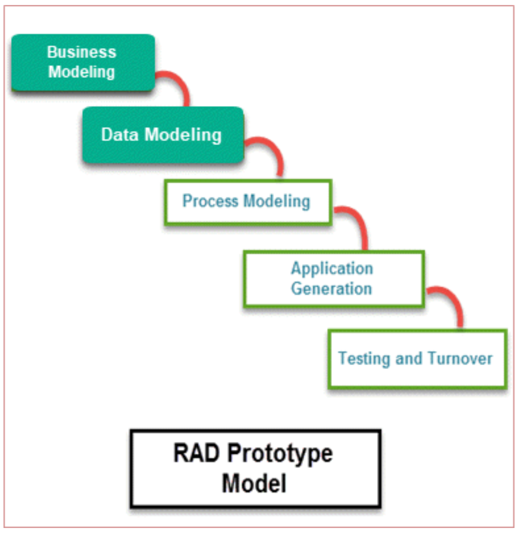
These models are depicted in Figure 3.1 below:

### Comparison of SDLC Models

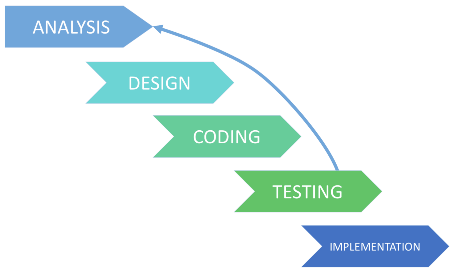
Table: 3.1 compares the General software process models on different Parameters as listed and compared in Sharma and Singh (2015).



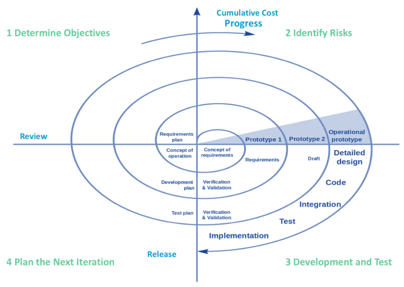
(a) Prototype Model, Source: Pal (2020)



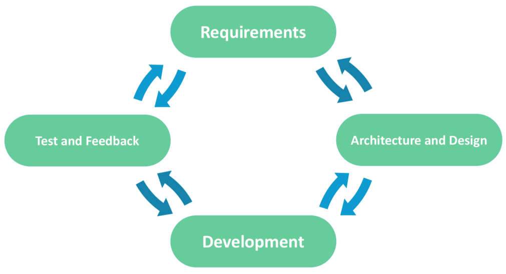
(b) Rapid application development model (RAD)



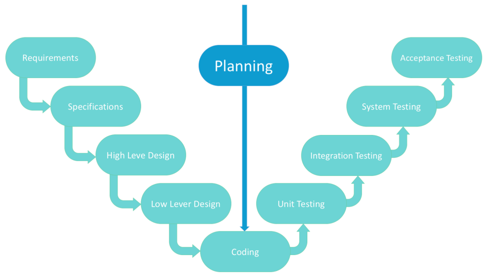
(c) Incremental model, Source: Team (2020)



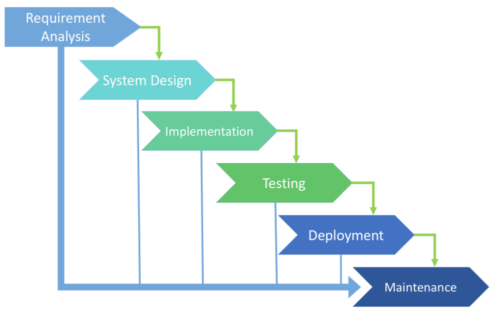
(d) Spiral model, Source: Team (2020)



(e) Agile model, Source: Team (2020)



(f) V-shaped model, Source: Team (2020)



(g) Waterfall model, Source: Team (2020)

Figure 3.1: SDLC Models

Table 3.1: Comparison of various SDLC Models on different Parameters

| **Model/Features** | **Waterfall** | **Prototype** | **RAD** | **Incremental** | **Spiral** | **Build & Fix** | **V-shaped** |
|---|---|---|---|---|---|---|---|
| Well defined requirements | Yes | No | Yes | No | No | No | Yes |
| User involvement in all phases | Only at beginning | High | Only at beginning | Yes(Intermediate) | High | No | No |
| Risk analysis | Only at beginning | No Risk analysis | Low | No Risk analysis | Yes | No | Only at beginning |
| Overlapping phases | No overlapping | Yes | No | No | Yes | Yes | No |
| Implementation time | Long | Quick | Quick | Long | Long | Depend upon Project | Long |
| Cost | Low | High | Low | Low | Expensive | Low | Expensive |
| Incorporation of changes | Difficult | Easy | Easy | Easy | Easy | Difficult | Difficult |
| Simplicity | Simple | Simple | Simple | Intermediate | Intermediate | Simple | Intermediate |
| Flexibility | Rigid | Little flexible | High | Less flexible | Flexible | FLexible | Less flexible |

## 3.1.2 Python

Python is the programming language in which Django and Flask frameworks, used in implementing majority of the projects, were written. It is a general-purpose, versatile and modern programming language with a dynamic scripting ability similar to Perl and Ruby. First released in 1991 by its principal author, Guido van Rossum, Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution, Zhou (2010).

## 3.1.3 Flask Framework

Flask is a lightweight Web Server Gateway Interface (WSGI) web application framework(Ronacher (2020)) written in Python by Armin Ronacher and first released on the 1st of April, 2010. It was designed as an extensible framework from the ground up providing a solid core with the basic services while allowing extensions to provide the rest. Since one can pick and choose the required extension packages, a lean stack with no bloat and which does exactly what one needs is ended up with.

Flask has two main dependencies, namely Werkzeug and Jinja2. Werkzeug provides the routing, debugging, and WSGI subsystems, while template support is provided by Jinja2 Grinberg (2014).

## 3.1.4 Django Framework

Django is a powerful Python web application framework that encourages rapid development with clean, and pragmatic design, which offers a relatively shallow learning curve, Melé (2018). It is open source with the primary goal of making the development of complex and data-based websites easier. Thus it emphasizes the re-usability and pluggability of components to ensure rapid developments, Zhou (2010). Django consists of three layers, namely model, view and template, Holovaty and Willison (2019).

#### Model Layer

Model, the single and definitive source of information about a data, contains the essential fields and behaviors of the data being stored. Generally, each model maps to a single database table, Holovaty and Willison (2019).

#### View Layer

Django's "views", which can be function- or class-based, encapsulate the logic that is responsible for processing a user's request and for returning the desired response, Holovaty and Willison (2019). Response may be an HyperText Markup Language (HTML) content, XML document or error code such as error 404, 505 and so on. The logic encapsulated in a view is allowed to be arbitrary provided that the desired response is returned.

#### Template Layer

The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted and A Django project can be configured with one or several template engines or systems such as the Django template language (DTL) and Jinja2.

### 3.1.5 AJAX

AJAX, a term coined in 2005 by Jesse James Garrett, stands for Asynchronous JavaScript and XML and refers to a set of web development techniques combining existing web technologies, including HTML or eXtensible HyperText Markup Language (XHTML), Cascading Style Sheets, JavaScript, The Document Object Model, XML, eXtensible Stylesheet Language Transformations (XSLT), and most importantly the XHR object, Yang (2020). Ajax is widely used in client-side programming (e.g. JavaScript) to allow for data to be sent and received to and from a database or server without actually disturbing the user experience or reloading the entire browser page. Figure 3.2 compares the conventional method of requesting data from a web server and the Ajax method.

AJAX was greatly utilized in carrying out my projects since most of them exhibited Real-time functionality. Code Snippet 3.1 is a sample snippet from one of the projects with asynchronous data persistence.

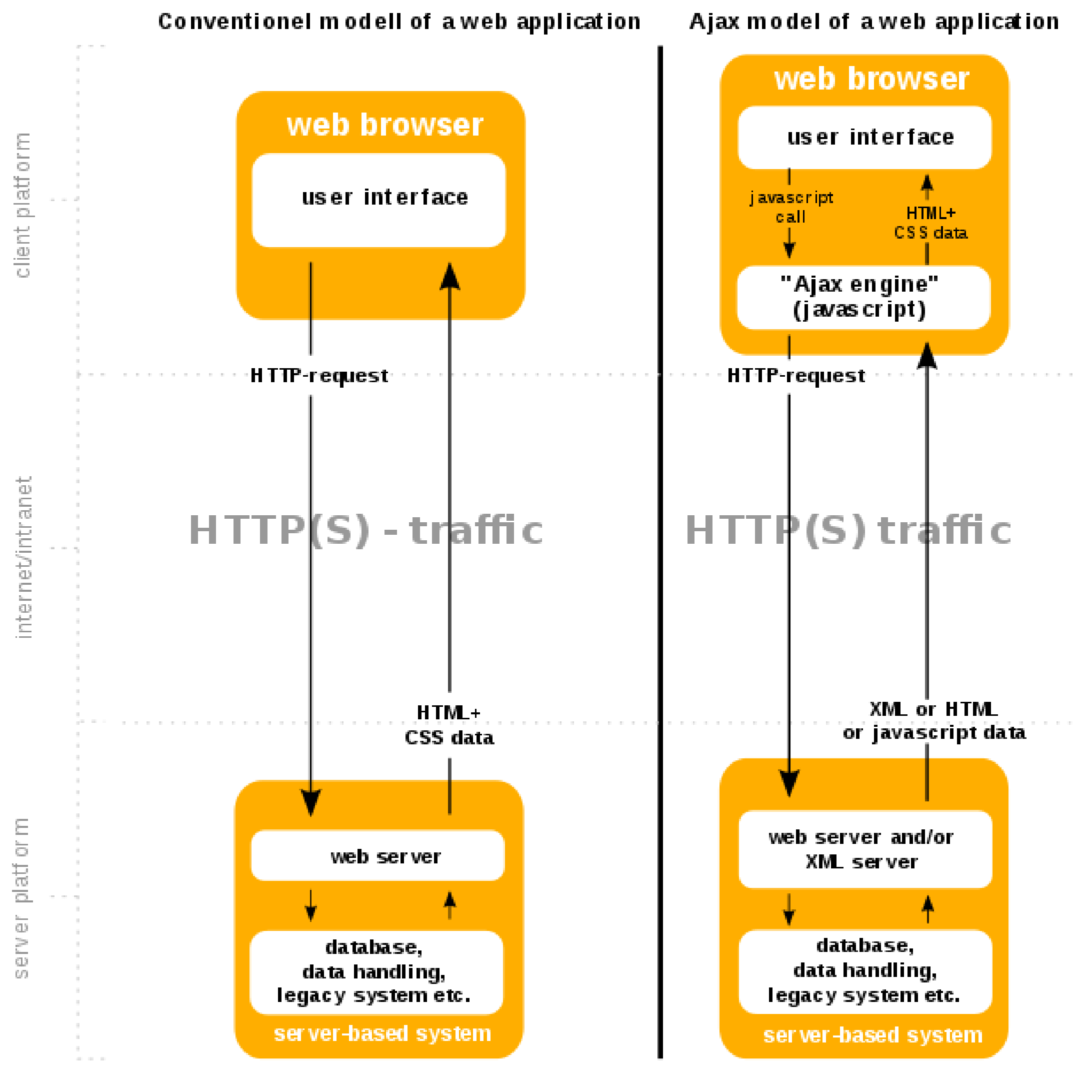


Figure 3.2: Conventional method vs AJAX method, Source Wikipedia.com

Code Snippet 3.1: Django Blog's AJAX snippet

```
$(document).ready(function(event) {
        $(document).on('submit', '#postcomment', function(event) {
        event.preventDefault();
        $.ajax({
                type: 'POST',
                url: $(this).attr('action'),
                data: $(this).serialize(),
                dataType: 'json',
                success: function(response) {
                        $('.comments-wrap').html(response['form']);
                                $(document).ready(function() {
                                        $("#commenters").on("click", ".reply",
                                        function(event) {
                                                event.preventDefault();
                                                var form = $("#postcomment").clone(true);
                                                form.find('.parent').val($(this)
                                                        .parent().parent().attr('id'));
                                                $(this).parent().append(form).fade();
                                });
                        });
                        $(document).ready(function() {
                                var ssPrettyPrint = function() {
                                        $('pre').addClass('prettyprint');
                                        $('code').addClass('prettyprint');
                                        $(document).ready(function() {
                                                PR.prettyPrint();
                                        });
                                };
                        });
                },
                error: function(rs, e) {
                        console.log(rs.responseText);
                },
        })
});
});
```

The above snippet was the underlying code responsible for the real-time functionality of Django blog's commenting system. It updates and fetches data from the Comment table which had been created in the blog's models without reloading the entire web page.

### 3.1.6 SQLAlchemy

According to Bayer (2016), The SQLAlchemy Structured Query Language (SQL) Toolkit and Object Relational Mapper (ORM) provide a comprehensive set of tools for working with databases and Python. It consists of several distinct areas of functionality which can be individually or collaboratively used together. Below is the illustration of the major components of SQLAlchemy, with component dependencies organized into layers:

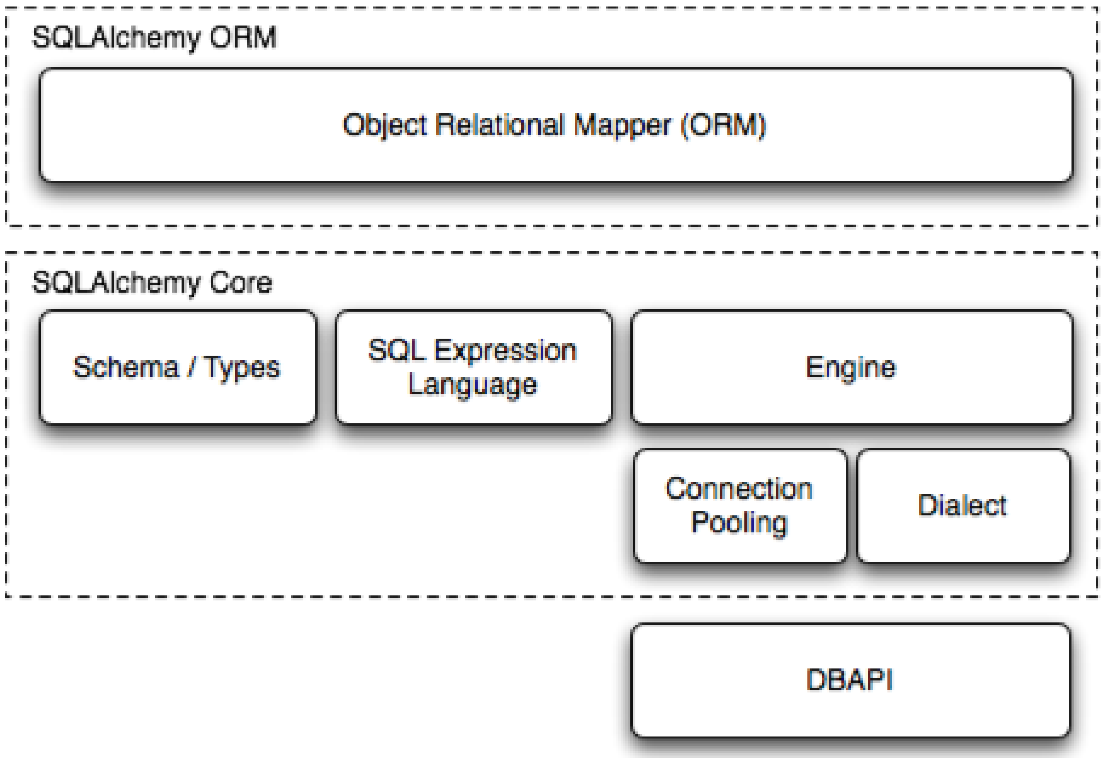


Figure 3.3: SQLAlchemy overview, from Bayer (2016)

From the above figure, the two most significant front-facing portions of SQLAlchemy are the ORM and the SQL Expression Language. SQL Expressions can be used independently of the ORM. When using the ORM, the SQL Expression language remains part of the public facing Application Programming Interface (API) as it is used within object-relational configurations and queries, Bayer (2016).

### 3.1.7 Jinja2

Jinja2 is a fast, expressive, extensible and modern day web templating language or engine for Python developers mimicking DTL with the inclusion of call functions with arguments on objects. From Ronacher (2017), An important usage of Jinga2 is the creation of HTML, XML or other markup formats which are returned to the user via HyperText Transfer Protocol (HTTP) response.

## 3.2 Python-based Projects

### 3.2.1 Methodology

All of the web applications were built using Flask or Django framework at the Back-end and Creative Tim's dashboard and other extensible templates at the front-end following the procedures below:

#### Created the Database tables

To get started, a back-end of the system which is the database was created. It should be noted that the implementation of this system is framework dependent as discussed below.

- **Flask**: One of the notable tables in the database was the User table which was created by writing a Class that inherited from Flask's model class. A snippet is as follows:

Code Snippet 3.2: vCPE Flask User Model

```python
from app         import db
from flask_login import UserMixin

from . common    import COMMON, STATUS, DATATYPE

class User(UserMixin, db.Model):

        id          = db.Column(db.Integer,     primary_key=True)
        user        = db.Column(db.String(64),  unique = True)
        email       = db.Column(db.String(120), unique = True)
        name        = db.Column(db.String(500))
        role        = db.Column(db.Integer)
        password    = db.Column(db.String(500))
        password_q  = db.Column(db.Integer)

        def __init__(self, user, password, name, email):
                self.user        = user
                self.password    = password
                self.password_q = DATATYPE.CRYPTED
                self.name        = name
                self.email       = email
                self.group_id = None
                self.role     = None

        def __repr__(self):
                return '<User %r>' % (self.id)

        def save(self):

                # inject self into db session
                db.session.add ( self )

                # commit change and save the object
                db.session.commit( )

                return self
```

This piece of code creates a table called `User` which stores the user information. The table has seven columns: `id`, a primary key; `user` which corresponds to `User`'s username; `email` corresponding to `User`'s email; `name` corresponding to the name of the `User`; role; `password` and `password_q` corresponding to the password and confirm password fields respectively of the `User`. There are also the `__init__`(Python's Object constructor), `__repr__`, and `save` methods all taking *"self"* as argument, a requirement in Python programming language for methods. `__init__` and `__repr__` are called *dunder* or Double Under(Underscores) methods which are some times referred to as Magic methods. When an object of the class `User` is created, the `__init__` method is invoked while `__repr__` as well as its counterpart, `__str__`, found in Code Snippet 3.2 line 31, controls the `to-string` conversion in the class that houses it.

It is worth noting that Flask uses `SQLAlchemy` for the ORM or Data Access part. It takes charge of the database management such as creating tables, writing SQL queries and other database operations.

- **Django**: For Django, creation of tables is fairly more verbose but simple. The snippet below buttresses this point.

Code Snippet 3.3: Django Blog's PublishedManager, Category,and Post

```
from django.contrib.postgres.fields import ArrayField
from django.db import models
from django.contrib.contenttypes.models import ContentType
from django.contrib.auth.models import User
from ckeditor_uploader.fields import RichTextUploadingField
from ckeditor.fields import RichTextField
from django.utils import timezone
from django.urls import reverse
from django.db.models.signals import pre_save
from PIL import Image
from taggit.managers import TaggableManager
from .utils import get_read_time
from django.utils.safestring import mark_safe
#from django.contrib.contenttypes.fields import GenericRelation

# Create your models here.

class PublishedManager(models.Manager):
        def get_queryset(self):
                return super(PublishedManager, self).get_queryset()
                .filter(status='published')

class Category(models.Model):
        name = models.TextField(max_length=1000000)
        slug = models.SlugField(max_length=10000000, unique=True)

        class Meta:
                ordering = ('name',)
                verbose_name='category'
                verbose_name_plural = 'categories'

        def __str__(self):
                return self.name

class Post(models.Model):
        STATUS_CHOICES = (
                ('draft', 'Draft'),
                ('published', 'Published'),
        )
        title   = models.CharField(max_length=10000000)
        slug    = models.SlugField(max_length=10000000, unique_for_date='publish')
        author  = models.ForeignKey(User, on_delete=models.CASCADE,
        related_name='blog_posts')
        image   = models.ImageField(upload_to='posts_pics'
        ,default='logo.svg')
        body    = RichTextUploadingField()
        publish = models.DateTimeField(default=timezone.now)
        read_time =  models.IntegerField(default=0)
        created = models.DateTimeField(auto_now_add=True)
        updated = models.DateTimeField(auto_now=True)
        status  = models.CharField(max_length=100
        ,choices=STATUS_CHOICES, default='draft')
        objects = models.Manager() # The default manager.
        published = PublishedManager() # Our custom manager.
        category = models.ForeignKey(Category, on_delete=models.CASCADE)
        tags = TaggableManager()
        class Meta:
                ordering = ('-created',)

        def __str__(self):
                return self.title


        def get_absolute_url(self):
                return reverse('blog:post_detail',args=[self.publish.year
                , self.publish.month, self.publish.day, self.slug])

        def get_next_post(self):
                return self.get_next_by_created()

        def get_previous_post(self):
                return self.get_previous_by_created()

        def get_markdown(self):
                body = self.body
                return mark_safe(body)

        @property
        def get_content_type(self):
                instance = self
                content_type = ContentType.objects.get_for_model(instance.__class__)
                return content_type

        def save(self, *args, **kwargs):
                super(Post, self).save(*args, **kwargs)

                img = Image.open(self.image.path)
                width = 883
                height = 391
                if img.width > width or img.height > height:
                        output_size = (width, height)
                        img.thumbnail(output_size)
                        img.save(self.image.path)
```

### Created the User Interface

Although the Database, where tables of any kind could be updated using Django and/or Flask models, had been created, it would be a disaster for users and the technical maintenance team if end users are allowed to manipulate the data directly. Therefore, nice and responsive user interfaces were created to let users interact with the data indirectly. As discussed above, Flask and Django provide templating components to create the user interfaces. Since Flask solely uses Jinja2 whereas Django has an optional DTL, the syntax for creating the user interfaces are a bit different.

- **Django**: Django supports two templating engines, namely Jinja2 and DTL where the latter happens to be the default with the below snippet.

Code Snippet 3.4: Django Blog's `post_form.html`

```
{% extends 'base.html' %}
{% block header %}{% endblock header %}
{% block content %}
<!-- site content
================================================== -->
<div class="s-content content">
        <main class="row content__page">
                <section class="column large-full entry format-standard">
                        <div class="content__page-header">
                                <h1 class="display-1">
                                        Write A Post.
                                </h1>
                        </div> <!-- end content__page-header -->
                        <form name="contactForm" id="contactForm" method="post"
                        action="" autocomplete="on" enctype="multipart/form-data">
                                {% csrf_token %}
                                <fieldset>
```

```
                                        {% for field in form %}
                                                <div class="form-field">
                                                        {{ form.media }}
                                                        {{ field.label }}
                                                        {{ field }}
                                                </div>
                                        {% endfor %}
                                        <input name="submit" id="submit" class="btn
                                        btn--small" value="Create Post" type="submit">
                                </fieldset>
                        </form> <!-- end form -->
                </section>
        </main>
</div> <!-- end s-content -->
{% endblock content %}
```

Every line surrounded by "% %" is a Django sentence which deals with simple "if-else", csrf_token and "for-loop" among others. The lines surrounded by double curly braces, such as {{ field }}, denote variables whose values are passed from view functions whenever this template has to be displayed.

- **Flask**: For flask, Jinja2 is one of the major dependencies needed to get it up and running and the syntax looks as follows:

Code Snippet 3.5: CEA Dashboard `login.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
{% if title %}
        <title>ipNX dashboard - {{ title }}</title>
{% else %}
        <title>ipNX dashboard</title>
{% endif %}
</head>
<body>
        <div class="limiter">
                <div class="container-login100">
                        <div class="wrap-login100">
                                <form class="login100-form validate-form"
                                method="POST" action="">
                                        {{ form.hidden_tag() }}
                                        <span class="login100-form-logo">
                                                <img
                                                        src="/static/assets/custom/login
                                                        /images/logo.png" alt="">
                                        </span>
                                        <span class="login100-form-title p-b-34 p-t-27">
                                                Log in
                                        </span>
                                        {% for message in get_flashed_messages() %}
                                                <div class="text-center p-t-5 p-b-5">
                                                        <p class="error">
                                                                {{ message }}
                                                        </p>
                                                </div>
                                        {% endfor %}
                                        <div class="wrap-input100 validate-input"
                                        data-validate="Enter E-mail">
                                                {{ form.email(class="input100",
                                                        placeholder="E-mail") }}
                                                <span class="focus-input100"
                                                        data-placeholder="&#xf207;">
                                                </span>
                                        </div>

                                        <div class="wrap-input100 validate-input"
                                        data-validate="Enter password">
                                                {{ form.password(placeholder="Password",
                                                        class="input100") }}
                                                        <span class="focus-input100"
                                                                data-placeholder="&#xf191;">
                                                        </span>
                                        </div>

                                        <div class="contact100-form-checkbox">
                                                {{ form.remember(class="input-checkbox100",
                                                        id="ckb1",type="checkbox") }}
                                                <label class="label-checkbox100" for="ckb1">
                                                        Remember me
                                                </label>
                                        </div>

                                        <div class="container-login100-form-btn">

                                                {{ form.submit(class="login100-form-btn") }}

                                        </div>

                                        <div class="text-center p-t-10">
                                                <a class="txt1" href="#">
                                                        Forgot Password?
                                                </a>
                                        </div>
                                </form>
                        </div>
                </div>
        </div>

</body>

</html>
```

### 3.2.2 Implemented the view function

Having dealt with the back-end databases and the frontend web pages user interfaces, the logic which stands in between to deal with the user requests and maintain the database were implemented. Unsurprisingly, the view systems in Flask and Django are different.

- **Django**: Django's views component provides a set of API to implement the logic. The Django view file, `views.py`, houses the functions and/or classes to achieve all set goals. The view functions and/or classes are shown in the snippet below:

Code Snippet 3.6: Django's Blog `views.py`

```
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.contenttypes.models import ContentType
from django.template.defaultfilters import slugify
from django.http import HttpResponse, HttpResponseRedirect,
Http404, JsonResponse
from django.core.paginator import Paginator,
EmptyPage, PageNotAnInteger
from django.views import generic
from django.contrib.auth.mixins import LoginRequiredMixin
, UserPassesTestMixin

from taggit.models import Tag
from django.contrib.auth.decorators import login_required
from django.db.models import Count, Q
from django.contrib.auth.models import User
from .models import Post, Comment
from .forms import PostForm, UpdatePostForm, CommentForm
from django.template.loader import render_to_string
from django.contrib.postgres.search import SearchVector,
SearchQuery, SearchRank, TrigramSimilarity

#Function based view
def blog_index(request, tag_slug=None):
        object_list = Post.published.all()
        if request.user.is_staff or request.user.is_superuser:
                object_list = Post.objects.all()
        common_tags = Post.tags.most_common()[:2]
        tag = None

        if tag_slug:
                tag = get_object_or_404(Tag, slug=tag_slug)
                object_list = object_list.filter(tags__in=[tag])

        paginator = Paginator(object_list, 5) # 5 posts in each page
        page = request.GET.get('page')
        try:
                posts = paginator.page(page)
                except PageNotAnInteger:
                # If page is not an integer deliver the first page
                posts = paginator.page(1)
        except EmptyPage:
                # If page is out of range deliver last page of results
                posts = paginator.page(paginator.num_pages)
        context = {
                'page': page,
                'posts': posts,
                'tag': tag,
                'common_tags': common_tags,
        }
        return render(request, 'blog/index.html', context)

#Class based view
class PostUpdateView(LoginRequiredMixin,
                UserPassesTestMixin, generic.UpdateView):
        model = Post
        form_class=UpdatePostForm
        template_name = 'blog/post_form.html'

        def form_valid(self, form):
                form.instance.author = self.request.user
                return super().form_valid(form)

        def test_func(self):
                post = self.get_object()
        if self.request.user == post.author or self.request.user.is_staff:
                return True
        return False
```

- **Flask**: According to Grinberg (2014), ”Clients such as web browsers send requests to the web server, which in turn sends them to the Flask application instance. The application instance needs to know what code needs to run for each URL requested, so it keeps a mapping of URLs to Python functions. The association between a URL and the function that handles it is called a `route`. The most convenient way to define a route in a Flask application is through the `app.route` decorator exposed by the application instance, which registers the decorated function as a `route`. The following example shows how a route is declared using this decorator:

  Code Snippet 3.7: Example snippet

```
@app.route('/')
def index():
        return '<h1>Hello World!</h1>'
```

  Functions like index() are called view functions” and are mostly housed in the `views.py` file and an example of a comprehensive `views.py` file is shown below:

  Code Snippet 3.8: An excerpt from CEA dashboard’s `views.py` file

```
# all the imports necessary

```

```
from flask import json, url_for, redirect, Markup,
Response, render_template, flash, g, session, jsonify,
request, send_from_directory
from werkzeug.exceptions import HTTPException, NotFound, abort

import os
import secrets
from PIL import Image
from app  import app

from flask       import url_for, redirect, render_template,
flash, g, session, jsonify, request, send_from_directory
from flask_login import login_user, logout_user,
current_user, login_required
from app         import app, lm, db, bc
from . models    import User
from . common    import COMMON, STATUS
from . assets    import *
from . forms     import LoginForm, RegisterForm, UpdateAccountForm
import os, shutil, re, cgi, json, random, time
from datetime import datetime


random.seed()  # Initialize the random number generator

# provide login manager with load_user callback
@lm.user_loader
def load_user(user_id):
        return User.query.get(int(user_id))
@app.route('/index')
@login_required
def index():
        inactiveCustomers = 34546
        activeCustomers = 7654984
        numberOfCalls = 123456
        numberOfRetailCustomers = 455674666
        return render_template('pages/index.html',
        numberOfCalls=numberOfCalls,
        numberOfRetailCustomers=numberOfRetailCustomers,
        inactiveCustomers=inactiveCustomers,
        activeCustomers=activeCustomers)


# authenticate user
@app.route('/logout')
def logout():
        logout_user()
        return redirect(url_for('login'))
```

### 3.2.3 Projects Implemented

**Customer Premises Equipment (CPE)**

- **Project's Requirement specification**: The project was to build a web UI for a customer premises equipment (CPE) through which settings can be modified.

  Please design such a web UI with the following pages:

  * Login page - username and password
  * Main page - device summary showing WAN and LAN IPv4 addresses
  * Settings page -

    Enable/disable DHCP service, modify address range, manage DHCP reservations

    Manage DNS servers

    Manage device whitelist/blacklist based on MAC addresses

    Manage WiFi settings including ESSID name and pre-shared key (password), enable/disable 2.4GHz and/or 5GHz bands, add or remove WiFi devices

  This must be a responsive site with a nice UI and interaction. Communication between front-end and back-end should JSON over XHR asynchronously. Backend should be Python or PHP based with data persistence in SQLite on the backend server.

Having implemented the features stated, the system was initiated and Figure 3.3 shows the pages of each feature:

### 3.2.4 Customer Experience Analysis (CEA) Dashboard

### 3.2.5 Data Usage Analytic (DUA) Dashboard

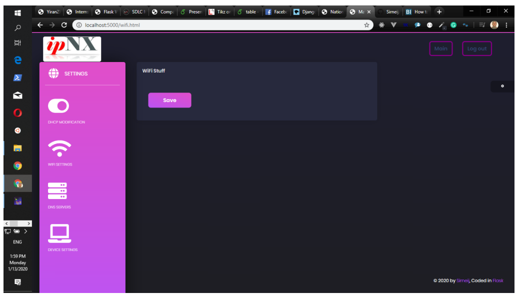
### 3.2.6 Flask Blog

### 3.2.7 Django Blog

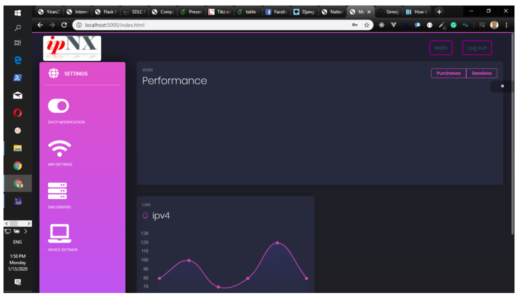
### 3.2.8 Web Crawler

### 3.2.9 Recommender System

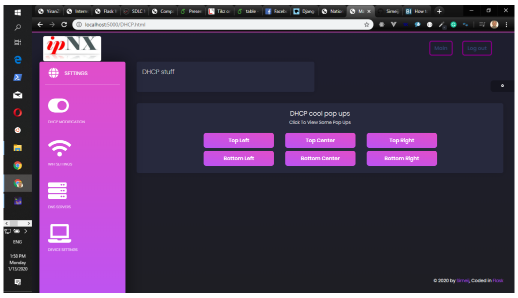
### 3.2.10 Deployment



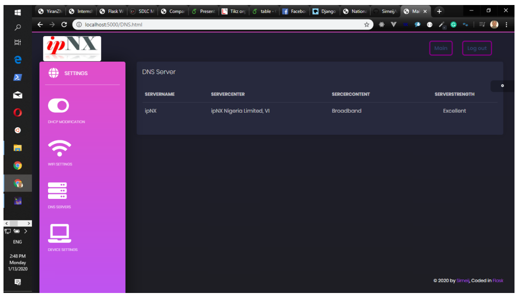
(a) vCPE WiFi management page



(b) vCPE Main page

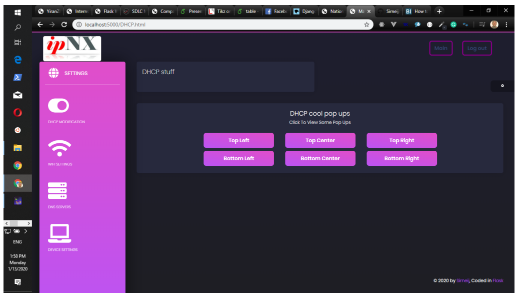


(c) vCPE DHCP page

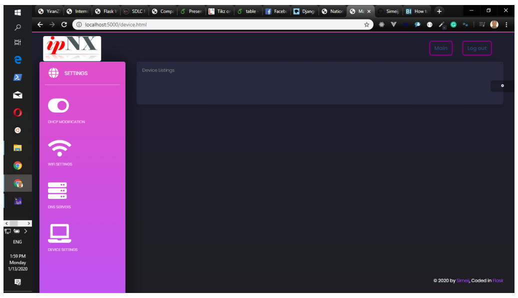


| SERVERNAME | SERVERCENTER | SERCERCONTENT | SERVERSTRENGTH |
|---|---|---|---|
| ipNX | ipNX Nigeria Limited, VI | Broadband | Excellent |

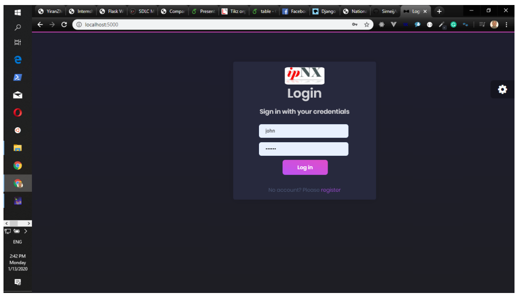
(d) vCPE DNS Server page



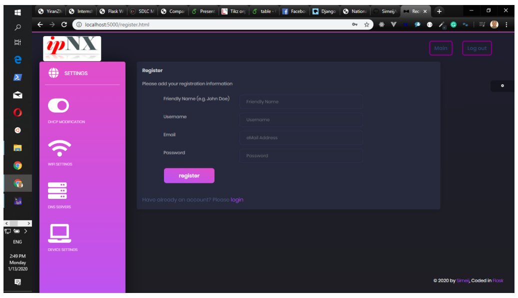
(e) vCPE DHCP page



(f) vCPE Devices List page



(g) vCPE Login page


localhost:5000/register.html
Register
Please add your registration information
Friendly Name (e.g. John Doe)
Friendly Name
Username
Username
Email
eMail Address
Password
Password
register
Have already an account? Please login

(h) vCPE Registration page, *restricted to an admin*.

Figure 3.4: vCPE Screenshots

# Chapter 4

# Conclusion and Recommendation

## 4.1 Conclusion

## 4.2 Recommendation

# References

Bayer, M. (2016). *SQLAlchemy Documentation*. Release 0.9.10.

Grinberg, M. (2014). *Flask Web Development*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, second edition.

Holovaty, A. and Willison, S. (2019). *Django Documentation*. Django Software Foundation. Release 3.1.dev.

Melé, A. (2018). *Django 2 by Example: Build powerful and reliable Python web applications from scratch*. Packt Publishing Ltd., 35, Livery Street, Birmingham B3 2PB, United Kingdom.

Pal, S. K. (2020 (accessed January 12, 2020)). *Software Engineering | Phases of Prototyping Model | Set - 2*. https://www.geeksforgeeks.org/software-engineering-phases-prototyping-model-set-2/.

Ronacher, A. (2010 (accessed January 9, 2020)). *Flask*. https://palletsprojects.com/p/flask/.

Ronacher, A. (2017). *Jinja2 Documentation*. Release 2.9.6.

Sharma, P. and Singh, D. (2015). Comparative study of various sdlc models on different parameters. *International Journal of Engineering Research*, 4(4):188–191.

Team, E. (2017 (accessed January 12, 2020)). *SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral*. https://existek.com/blog/sdlc-models/.

Yang, J. (2020 (accessed January 13, 2020)). *Ajax*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

Zhou, Y. (2010). Project report of building course management system by django framework. Report P-5, Simon Fraser University.