**Maharaja Surajmal Institute of Technology**

**New Delhi - 110058**

****

**Summer Training Report**

**On**

**Fake News Detector**

Submitted to Guru Gobind Singh Indrapastha University, Delhi (India) in partial fulfilment of the requirement for the award of the degree of

**B.Tech**

**in**

**Information Technology Submitted By**

**Ashish Dagar**

**Enrollment No. 00696303116**

**Acknowledgement**

A project work owes its success from commencement to completion to the people in love with project at various stages. Let me in this page express my gratitude to all those who helped in various stages. First I would like to express my sincere gratitude indebtness to **Mr. Manoj Malik** (HOD, Department of Information Technology) & **Mr. Satender Malik** (Proctor of IT Evening Shift) for allowing me to undergo the summer training of 6 weeks at OSAHUB, MSIT.

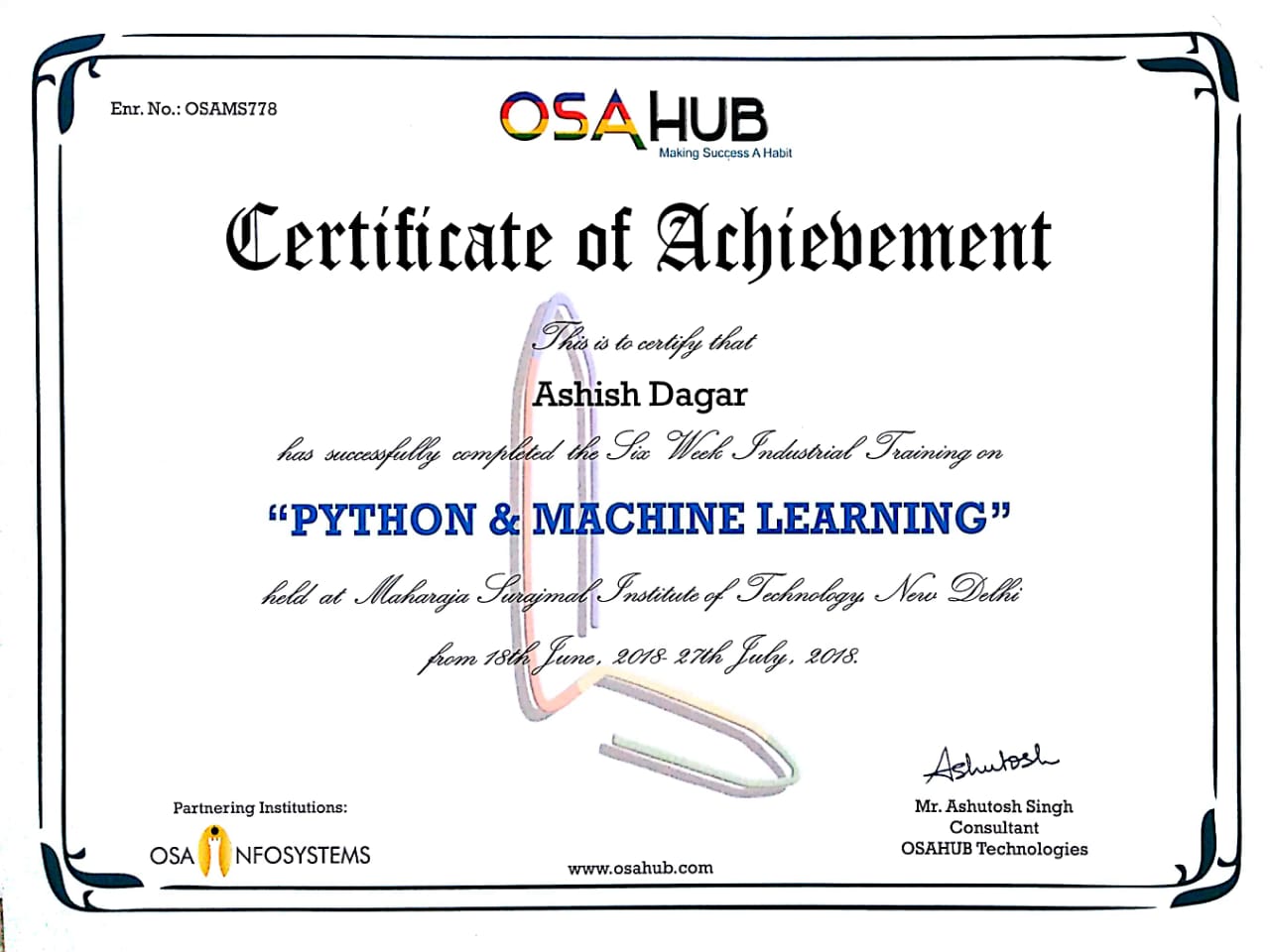
I am grateful to our guide **Mr. Ashutosh Singh** for the help provided in completion of the project, which was assigned to me. Whithout his friendly help and guidance it was difficult to develop this project.

Last but not least, I pay my sincere thanks and gratitude to all the staff members of **OSAHUB** for their support and for making our training valuable and fruitful.

Submitted By: Ashish Dagar

00696303116

**Certificate**

****

**Candidate’s Declaration**

I Ashish Dagar, Enrollment No. 00696303116, B.Tech (Semester – 5th) of the Maharaja Surajmal Institute of Technology, New Delhi hereby declare that the training report entitled “**Fake News Detector**” is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any other degree.

Ashish Dagar

01696302715

**Company Profile**

C:\Users\himalaya\Downloads\osahubLogo.png

OSAHUB Technologies is India's leading training provider. We focus on education solutions, covering all the latest technologies - be it in the field of **Computer Science, Electronics, Mechanical Engineering, Robotics, Animation, etc.** We are Ranked among Top Training Institutions / Education Centres and Graded among India's Most Trusted Service Brands. We provide Workshops, Trainings, Faculty Development Programs, Certifications Courses, Seminars, Career Counselling, etc. to engineering colleges and schools all across India. We have an extremely skilled network of subject matter experts and highly experienced trainers who have several years of experience in the industry. We are pioneers in the education industry and are proud of our excellent team.

Our vision is to act as a mediator between Technology and Students to improve their academic performance by providing non-syllabus inputs and Best Trainings on various Emerging Technologies. Our vision is to empower youth through high quality and dedicated education.

**Introduction to Resource Grabber**

Resource Grabber is basically a shopping web-application. As you can see in your college’s photocopy shop paper is wasted like water, by giving same notes every year. Any user who is registered in this website can upload their used resources like drafter, photocopied notes, handmade notes and reference books etc and junior students of same course can see these product in this website then these students can contact you for buying these items if they need. Senior students can also sell their resources to junior students for free as per their desired wish. Names of the reference books are provided in this website. Students can also search the name of the resource or item in shop page.

You can register in this website for free or there is no cost. By subscribing in this website you will get notification of the recently uploaded items or resources through E-mail.

The main concern of this website is to avoid wastage of papers or resources and junior students can avail these resources.

**Contents**

**Title Page i Acknowledgement ii Certificate iii Candidate’s Declaration iv Company Profile v Introduction to Resource Grabber vi**

**Chapter 1**

**1.1 Introduction to Website ix**

**1.2 Technology Used x**

**1.2.1 HTML x**

**1.2.2 CSS xiv**

**1.2.3 JavaScript xvi**

**1.2.4 Bootstrap Themes xviii**

**1.2.5 JAVA xx**

**1.2.6 JAVA Servlet xxii**

**1.2.8 XML xxiv**

**1.2.8 JSP xv**

**1.2.9 Google Cloud DataStore xvii**

**1.3 Tools Used xxix**

**1.3.1 Eclipse IDE xxix**

**1.3.2 Google App Engine xxxii**

**1.3.3 Atom xxxiv**

**1.3.3 Amazon Web Services xxxv**

**Chapter 2**

**2.1 Introduction xxxix**

**2.2 Project Aim and Objectives xxxix**

**2.3 Operational Environment xl**

**2.3.1 Minimum Requirements xl**

**2.3.2 Recommended Requirements xl**

**2.4 System Aim xl**

**2.5 System Design xl**

**2.5.1 Table Design xl**

**2.6 Data Flow Diagram xli**

**2.7 Web-Applicaltion Implementation xlii**

**2.7.1 Screenshots xlii**

**Bibliography xlviii**

**Chapter 1**

**1.1 Introduction to Website**

A **website**, or simply a **site**, is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server. A website may be accessible via a public Internet Protocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing a uniform resource locator (URL) that identifies the site.

Websites have many functions and can be used in various fashions; a website can be a personal website, a commercial website for a company, a government website or a non-profit organization website. Websites are typically dedicated to a particular topic or purpose, ranging from entertainment and social networking to providing news and education. All publicly accessible websites collectively constitute the World Wide Web, while private websites, such as a company's website for its employees, are typically a part of an intranet.

Web pages, which are the building blocks of websites, are documents, typically composed in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML, XHTML). They may incorporate elements from other websites with suitable markup anchors. Web pages are accessed and transported with the Hypertext Transfer Protocol (HTTP), which may optionally employ encryption (HTTP Secure, HTTPS) to provide security and privacy for the user. The user's application, often a web browser, renders the page content according to its HTML markup instructions onto a display terminal.

Hyperlinking between web pages conveys to the reader the site structure and guides the navigation of the site, which often starts with a home page containing a directory of the site web content. Some websites require user registration or subscription to access content. Examples of subscription websites include many business sites, news websites, academic journal websites, gaming websites, file-sharing websites, message boards, web-based email, social networking websites, websites providing real-time stock market data, as well as sites providing various other services. As of 2016 end users can access websites on a range of devices, including desktop and laptop computers, tablet computer, smartphones and smart TVs.

**History**

The World Wide Web (WWW) was created in 1990 by the British CERN physicist Tim Berners-Lee. On 30 April 1993, CERN announced that the World Wide Web would be free to use for anyone. Before the introduction of HTML and HTTP, other protocols such as File Transfer Protocol and the gopher protocol were used to retrieve individual files from a server. These protocols offer a simple directory structure which the user navigates and chooses files to download. Documents were most often presented as plain text files without formatting, or were encoded in word processor formats.

**Overview**

Websites have many functions and can be used in various fashions; a website can be a personal website, a commercial website, a government website or a non-profit organization website. Websites can be the work of an individual, a business or other organization, and are typically dedicated to a particular topic or purpose. Any website can contain a hyperlink to any other website, so the distinction between individual sites, as perceived by the user, can be blurred. Websites are written in, or converted to, HTML (Hyper Text Markup Language) and are accessed using a software interface classified as a user agent. Web pages can be viewed or otherwise accessed from a range of computer-based and Internet-enabled devices of various sizes, including desktop computers, laptops, tablet computers and smartphones. A website is hosted on a computer system known as a web server, also called an HTTP (Hyper Text Transfer Protocol) server. These terms can also refer to the software that runs on these systems which retrieves and delivers the web pages in response to requests from the website's users. Apache is the most commonly used web server software (according to Netcraft statistics) and Microsoft's IIS is also commonly used. Some alternatives, such as Nginx, Lighttpd, Hiawatha or Cherokee, are fully functional and lightweight.

**Static Website**

A static website is one that has web pages stored on the server in the format that is sent to a client web browser. It is primarily coded in [Hypertext Markup Language](https://en.wikipedia.org/wiki/Hypertext_Markup_Language) (HTML); [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) are used to control appearance beyond basic HTML. Images are commonly used to effect the desired appearance and as part of the main content. Audio or video might also be considered "static" content if it plays automatically or is generally non-interactive. This type of website usually displays the same information to all visitors. Similar to handing out a printed brochure to customers or clients, a static website will generally provide consistent, standard information for an extended period of time. Although the website owner may make updates periodically, it is a manual process to edit the text, photos and other content and may require basic website design skills and software. Simple forms or marketing examples of websites, such as *classic website*, a *five-page website* or a *brochure website* are often static websites, because they present pre-defined, static information to the user. This may include information about a company and its products and services through text, photos, animations, audio/video, and navigation menus.

Static websites can be edited using four broad categories of software:

* [Text editors](https://en.wikipedia.org/wiki/Text_editor), such as [Notepad](https://en.wikipedia.org/wiki/Notepad_(Windows)) or [TextEdit](https://en.wikipedia.org/wiki/TextEdit" \o "TextEdit), where content and HTML markup are manipulated directly within the editor program
* [WYSIWYG](https://en.wikipedia.org/wiki/WYSIWYG) offline editors, such as [Microsoft FrontPage](https://en.wikipedia.org/wiki/Microsoft_FrontPage) and [Adobe Dreamweaver](https://en.wikipedia.org/wiki/Adobe_Dreamweaver) (previously Macromedia Dreamweaver), with which the site is edited using a [GUI](https://en.wikipedia.org/wiki/GUI) and the final HTML markup is generated automatically by the editor software
* WYSIWYG online editors which create media rich online presentation like web pages, widgets, intro, blogs, and other documents.
* [Template-based editors](https://en.wikipedia.org/wiki/Web_template_system) such as [iWeb](https://en.wikipedia.org/wiki/IWeb" \o "IWeb) allow users to create and upload web pages to a web server without detailed HTML knowledge, as they pick a suitable template from a palette and add pictures and text to it in a [desktop publishing](https://en.wikipedia.org/wiki/Desktop_publishing) fashion without direct manipulation of HTML code.

Static websites may still use server side includes (SSI) as an editing convenience, such as sharing a common menu bar across many pages. As the site's behaviour *to the reader* is still static, this is not considered a dynamic site.

**Dynamic Website**

A dynamic website is one that changes or customizes itself frequently and automatically. Server-side dynamic pages are generated "on the fly" by computer code that produces the HTML (CSS are responsible for appearance and thus, are static files). There are a wide range of software systems, such as CGI, Java Servlets and Java Server Pages (JSP), Active Server Pages and ColdFusion (CFML) that are available to generate dynamic web systems and dynamic sites. Various web application frameworks and web template systems are available for general-use programming languages like Perl, PHP, Python and Ruby to make it faster and easier to create complex dynamic websites.

A site can display the current state of a dialogue between users, monitor a changing situation, or provide information in some way personalized to the requirements of the individual user. For example, when the front page of a news site is requested, the code running on the web server might combine stored HTML fragments with news stories retrieved from a database or another website via RSS to produce a page that includes the latest information. Dynamic sites can be interactive by using HTML forms, storing and reading back browser cookies, or by creating a series of pages that reflect the previous history of clicks. Another example of dynamic content is when a retail website with a database of media products allows a user to input a search request, e.g. for the keyword Beatles. In response, the content of the web page will spontaneously change the way it looked before, and will then display a list of Beatles products like CDs, DVDs and books. Dynamic HTML uses JavaScript code to instruct the web browser how to interactively modify the page contents. One way to simulate a certain type of dynamic website while avoiding the performance loss of initiating the dynamic engine on a per-user or per-connection basis, is to periodically automatically regenerate a large series of static pages.

**Types**

Websites can be divided into two broad categories—static and interactive. Interactive sites are part of the Web 2.0 community of sites, and allow for interactivity between the site owner and site visitors or users. Static sites serve or capture information but do not allow engagement with the audience or users directly. Some websites are informational or produced by enthusiasts or for personal use or entertainment. Many websites do aim to make money, using one or more business models, including:

• Posting interesting content and selling contextual advertising either through direct sales or through an advertising network.

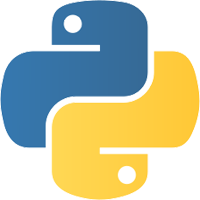
• E-commerce: products or services are purchased directly through the website

• Advertising products or services available at a brick and mortar business

• Freemium: basic content is available for free but premium content requires a payment (e.g., WordPress website, it is an open source platform to build a blog or website.)

**1.2 Technology Used**

**1.2.1 PYTHON**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python has a design philosophy that emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability), notably using [significant whitespace](https://en.wikipedia.org/wiki/Significant_whitespace). It provides constructs that enable clear programming on both small and large scales.[]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-7-27) In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).[[30]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-About-30)

Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source)software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation).

HISTORY

Python was conceived in the late 1980s, and its implementation began in December 1989 by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)) (itself inspired by [SETL](https://en.wikipedia.org/wiki/SETL)) capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system  Van Rossum remains Python's principal author. His continuing central role in Python's development is reflected in the title given to him by the Python community: [*Benevolent Dictator For Life*](https://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life) (BDFL) – a post from which he gave himself permanent vacation on July 12, 2018.

On the origins of Python, Van Rossum wrote in 1996:

...In December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)) that would appeal to [Unix](https://en.wikipedia.org/wiki/Unix)/[C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language))[hackers](https://en.wikipedia.org/wiki/Hacker_(programmer_subculture)). I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of [*Monty Python's Flying Circus*](https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus)).

— Guido van Rossum

Python 2.0 was released on 16 October 2000 and had many major new features, including a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode). With this release, the development process became more transparent and community-backed.

Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility) with previous versions. However, many of its major features have been [backported](https://en.wikipedia.org/wiki/Backporting" \o "Backporting) to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

Python 2.7's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)) date was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to [Go](https://en.wikipedia.org/wiki/Go_(programming_language)) [transcompiler](https://en.wikipedia.org/wiki/Transcompiler" \o "Transcompiler) to improve performance under concurrent workloads.

## Features and philosophy

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming" \o "Metaprogramming) and [metaobjects](https://en.wikipedia.org/wiki/Metaobject" \o "Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). It also features dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages))([late binding](https://en.wikipedia.org/wiki/Late_binding)), which binds method and variable names during program execution.

Python's design offers some support for [functional programming](https://en.wikipedia.org/wiki/Functional_programming) in the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition. It has filter(), map(), and reduce() functions; [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), and sets; and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).

The language's core philosophy is summarized in the document *The*[*Zen of Python*](https://en.wikipedia.org/wiki/Zen_of_Python) (*PEP 20*), which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:

* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of [Perl](https://en.wikipedia.org/wiki/Perl)) in favor of a simpler, less-cluttered grammar. As [Alex Martelli](https://en.wikipedia.org/wiki/Alex_Martelli) put it: "To describe something as 'clever' is *not* considered a compliment in the Python culture." Python's philosophy rejects the Perl "[there is more than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use [PyPy](https://en.wikipedia.org/wiki/PyPy" \o "PyPy), a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython" \o "Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group [Monty Python](https://en.wikipedia.org/wiki/Monty_Python)—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a [famous Monty Python sketch](https://en.wikipedia.org/wiki/Spam_(Monty_Python))) instead of the standard [foo and bar](https://en.wikipedia.org/wiki/Foobar" \o "Foobar).

A common [neologism](https://en.wikipedia.org/wiki/Neologism) in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonists*, *Pythonistas*, and *Pythoneers*.



**1.2.2** Jupyter Notebook

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface) (Formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter notebooks document can be converted to a number of [open standard](https://en.wikipedia.org/wiki/Open_standard) output formats ([HTML](https://en.wikipedia.org/wiki/HTML), [presentation slides](https://en.wikipedia.org/wiki/Presentation_slide), [LaTeX](https://en.wikipedia.org/wiki/LaTeX" \o "LaTeX), [PDF](https://en.wikipedia.org/wiki/PDF), [ReStructuredText](https://en.wikipedia.org/wiki/ReStructuredText" \o "ReStructuredText), [Markdown](https://en.wikipedia.org/wiki/Markdown), [Python](https://en.wikipedia.org/wiki/Python_(programming_language))) through 'Download As' in the web interface, via the [nbconvert](https://nbconvert.readthedocs.io/) library or 'jupyter nbconvert' command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through [NbViewer](https://nbviewer.org/) which can take a URL to any publicly available notebook document, convert it to HTML on the file and display it to the user.

Jupyter Notebook provides a browser-based [REPL](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop) built upon a number of popular [open-source](https://en.wikipedia.org/wiki/Open-source_software) libraries:

* [IPython](https://en.wikipedia.org/wiki/IPython)
* [ØMQ](https://en.wikipedia.org/wiki/%C3%98MQ)
* [Tornado (web server)](https://en.wikipedia.org/wiki/Tornado_(web_server))
* [jQuery](https://en.wikipedia.org/wiki/JQuery)
* [Bootstrap (front-end framework)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
* [MathJax](https://en.wikipedia.org/wiki/MathJax)

Jupyter Notebook can connect to many kernels, (by default Jupyter Notebook ships with the IPython kernel) to allow programming in many languages. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language))

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple](https://en.wikipedia.org/wiki/Maple_(software)), [Mathematica](https://en.wikipedia.org/wiki/Mathematica" \o "Mathematica), and [SageMath](https://en.wikipedia.org/wiki/SageMath" \o "SageMath), a computational interface style that originated with Mathematica in the 1980s. According to [The Atlantic](https://en.wikipedia.org/wiki/The_Atlantic), Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

### Jupyter kernels

A Jupyter kernel is a program responsible for handling various types of request (code execution, code completions, inspection), and providing a reply. Kernels talks to the other components of Jupyter using [ZeroMQ](https://en.wikipedia.org/wiki/ZeroMQ" \o "ZeroMQ) over the network, and thus can be on the same or remote machines. Unlike many other Notebook-like interface, in Jupyter, kernels are not aware they are attached to a specific document, and can be connected to from many client at once. Usually Kernels are implemented and allow execution of a single language with a couple of exceptions.

By default Jupyter ships with IPython as a default kernel and reference implementation via the ipykernel wrapper. Kernels of various quality and features for many languages are [available](https://github.com/jupyter/jupyter/wiki/Jupyter-kernels).

### Jupyter Hub

JupyterHub is a multi-user server for Jupyter Notebooks. It is designed to support many users by spawning, managing, and proxying many singular Jupyter Notebook servers.

### Jupyter Lab

JupyterLab is the next-generation user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful user interface. The first stable release was announced on February 20, 2018.

The Jupyter Notebook has become a popular user interface for [Cloud computing](https://en.wikipedia.org/wiki/Cloud_computing), and major cloud providers have adopted the Jupyter Notebook or derivative tools as a frontend interface for cloud users. Examples include [Amazon's](https://en.wikipedia.org/wiki/Amazon_(company)) SageMaker Notebooks, [Google's](https://en.wikipedia.org/wiki/Google) [Colaboratory](https://en.wikipedia.org/wiki/Colaboratory" \o "Colaboratory), and [Microsoft's](https://en.wikipedia.org/wiki/Microsoft) Azure Notebook

**1.2.3 JavaScript**

**** JavaScript, often abbreviated as JS, is a high-level, dynamic, weakly typed, object-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMAScript specification, with some engines not supporting the spectrum fully, and with many engines supporting additional features beyond ECMA.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as Self and Scheme. some engines not supporting the spectrum fully, and with many engines supporting additional features beyond ECMA.

**History**

In 1993, the National Center for Supercomputing Applications (NCSA), a unit of the University of Illinois at Urbana-Champaign, released NCSA Mosaic, the first popular graphical Web browser, which played an important part in expanding the growth of the nascent World Wide Web. In 1994, a company called Mosaic Communications was founded in Mountain View, California and employed many of the original NCSA Mosaic authors to create Mosaic Netscape. However, it intentionally shared no code with NCSA Mosaic. The internal codename for the company's browser was Mozilla, which stood for "Mosaic killer", as the company's goal was to displace NCSA Mosaic as the world's number one web browser. The first version of the Web browser, Mosaic Netscape 0.9, was released in late 1994. Within four months it had already taken three-quarters of the browser market and became the main browser for the Internet in the 1990s. To avoid trademark ownership problems with the NCSA, the browser was subsequently renamed Netscape Navigator in the same year, and the company took the name Netscape Communications.

Netscape Communications realized that the Web needed to become more dynamic. Marc Andreessen, the founder of the company believed that HTML needed a "glue language" that was easy to use by Web designers and part-time programmers to assemble components such as images and plugins, where the code could be written directly in the Web page markup. In 1995, the company recruited Brendan Eich with the goal of embedding the Scheme programming language into its Netscape Navigator. Before he could get started, Netscape Communications collaborated with Sun Microsystems to include in Netscape Navigator Sun's more static programming language Java, in fries to compete with Microsoft for user adoption of Web technologies and platforms. Netscape Communications then decided that the scripting language they wanted to create would complement Java and should have a similar syntax, which excluded adopting other languages such as Perl, Python, TCL, or Scheme. To defend the idea of JavaScript against competing proposals, the company needed a prototype. Eich wrote one in 10 days, in May 1995.

Although it was developed under the name Mocha, the language was officially called LiveScript when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed JavaScript when it was deployed in the Netscape Navigator 2.0 beta 3 in December. The final choice of name caused confusion, giving the impression that the language was a spin-off of the Java programming language, and the choice has been characterized as a marketing ploy by Netscape to give JavaScript the cachet of what was then the hot new Web programming language.

There is a common misconception that JavaScript was influenced by an earlier Web page scripting language developed by Nombas named C-- (not to be confused with the later C-- created in 1997). Brendan Eich, however, had never heard of C-- before he created LiveScript. Nombas did pitch their embedded Web page scripting to Netscape, though Web page scripting was not a new concept, as shown by the ViolaWWW Web browser. Nombas later switched to offering JavaScript instead of C-- in their ScriptEase product and was part of the TC39 group that standardized ECMAScript.

Microsoft script technologies including VBScript and JScript were released in 1996. JScript, a reverse-engineered implementation of Netscape's JavaScript, was part of Internet Explorer 3. JScript was also available for server-side scripting in Internet Information Server. Internet Explorer 3 also included Microsoft's first support for CSS and various extensions to HTML, but in each case the implementation was noticeably different to that found in Netscape Navigator at the time. These differences made it difficult for designers and programmers to make a single website work well in both browsers, leading to the use of "best viewed in Netscape" and "best viewed in Internet Explorer" logos that characterized these early years of the browser wars. JavaScript began to acquire a reputation for being one of the roadblocks to a cross-platform and standards-driven Web. Some developers took on the difficult task of trying to make their sites work in both major browsers, but many could not afford the time. With the release of Internet Explorer 4, Microsoft introduced the concept of Dynamic HTML, but the differences in language implementations and the different and proprietary Document Object Models remained and were obstacles to widespread take-up of JavaScript on the Web.

**1.2.4 Bootstrap Themes**

**** Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Bootstrap is the second most-starred project on GitHub, with more than 111,600 stars and 51,500 forks.

Bootstrap is the second most-starred project on GitHub, with more than 111,600 stars and 51,500 forks.

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to Twitter developer Mark Otto:

"A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company."

After a few months of development by a small group, many developers at Twitter began to contribute to the project as a part of Hack Week, a hackathon-style week for the Twitter development team. It was renamed from Twitter Blueprint to Bootstrap, and released as an open source project on August 19, 2011. It has continued to be maintained by Mark Otto, Jacob Thornton, and a small group of core developers, as well as a large community of contributors.

On January 31, 2012, Bootstrap 2 was released, which added a twelve-column responsive grid layout system, inbuilt support for Glyphicons, several new components, as well as changes to many of the existing components.

On August 19, 2013, Bootstrap 3 was released, which redesigned components to use flat design, and a mobile first approach.

On October 29, 2014, Mark Otto announced that Bootstrap 4 was in development. The first alpha version of Bootstrap 4 was released on August 19, 2015. The first beta version was released on 10 August 2017.

Bootstrap 3 supports the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera, and Safari (except on Windows). It additionally supports back to IE8 and the latest Firefox Extended Support Release (ESR).

Since 2.0, Bootstrap supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone).

Starting with version 3.0, Bootstrap adopted a mobile-first design philosophy, emphasizing responsive design by default.

The version 4.0 alpha release added Sass and flexbox support.

Bootstrap is modular and consists of a series of Less stylesheets that implement the various components of the toolkit. These stylesheets are generally compiled into a bundle and included in web pages, but individual components can be included or removed. Bootstrap provides a number of configuration variables that control things such as color and padding of various components.

Since Bootstrap 2, the Bootstrap documentation has included a customization wizard which generates a customized version of Bootstrap based on the requested components and various settings.

As of Bootstrap 4, Sass is used instead of Less for the stylesheets.

Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code.

Grid system and responsive design comes standard with an 1170 pixel wide grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones, portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

**1.2.5 JAVA**

****Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

The latest version is Java 9, released on September 21, 2017,[22] and is one of the two versions currently supported for free by Oracle. Versions earlier than Java 8 are supported both by Oracle and other companies on a commercial basis.

**History**

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time. The language was initially called Oak after an oak tree that stood outside Gosling's office. Later the project went by the name Green and was finally renamed Java, from Java coffee. Gosling designed Java with a C/C++-style syntax that system and application programmers would find familiar.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. The Java 1.0 compiler was re-written in Java by Arthur van Hoff to comply strictly with the Java 1.0 language specification. With the advent of Java 2 (released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. J2EE included technologies and APIs for enterprise applications typically run in server environments, while J2ME featured APIs optimized for mobile applications. The desktop version was renamed J2SE. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC 1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a de facto standard, controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine (JVM) as free and open-source software, (FOSS), under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of its JVM's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.

Sun's vice-president Rich Green said that Sun's ideal role with regard to Java was as an "evangelist".[34] Following Oracle Corporation's acquisition of Sun Microsystems in 2009–10, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency". This did not prevent Oracle from filing a lawsuit against Google shortly after that for using Java inside the Android SDK (see Google section below). Java software runs on everything from laptops to data centers, game consoles to scientific supercomputers.[36] On April 2, 2010, James Gosling resigned from Oracle.

In January 2016, Oracle announced that Java runtime environments based on JDK 9 will discontinue the browser plugin.

**1.2.6 JAVA Servlet**

A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

A Java servlet processes or stores a Java class in Java EE that conforms to the Java Servlet API, a standard for implementing Java classes that respond to requests. Servlets could in principle communicate over any client–server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of the web container and a servlet.

A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the web server and a client. Servlets may be packaged in a WAR file as a web application.

Servlets can be generated automatically from JavaServer Pages (JSP) by the JavaServer Pages compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. While the direct usage of servlets to generate HTML (as shown in the example below) has become rare, the higher level MVC web framework in Java EE (JSF) still explicitly uses the servlet technology for the low level request/response handling via the FacesServlet. A somewhat older usage is to use servlets in conjunction with JSPs in a pattern called "Model 2", which is a flavor of the model–view–controller.

The current version of Servlet is 3.1. This is the upgraded version.



The Servlet1 specification was created by Pavni Diwanji while she worked at Sun Microsystems, with version 1.0 finalized in June 1997. Starting with version 2.2, the specification was developed under the Java Community Process. As of September 2017, the current version of the Servlet specification is 4.0.

In his blog on java.net, Sun veteran and GlassFish lead Jim Driscoll details the history of servlet technology. James Gosling first thought of servlets in the early days of Java, but the concept did not become a product until Sun shipped the Java Web Server product. This was before what is now the Java Platform, Enterprise Edition was made into a specification.

The advantages of using servlets are their fast performance and ease of use combined with more power over traditional CGI (Common Gateway Interface). Traditional CGI scripts written in Java have a number of performance disadvantages:

When an HTTP request is made, a new process is created each time the CGI script is called. The overhead associated with process creation can dominate the workload especially when the script does relatively fast operations. Thus, process creation will take more time for CGI script execution. In contrast, for servlets, each request is handled by a separate Java thread within the web server process, thereby avoiding the overhead associated with forking processes within the HTTP daemon.

Simultaneous CGI requests will load the CGI script to be copied into memory once per request. With servlets, there is only one copy that persists across requests and is shared between threads.

Only a single instance answers all requests concurrently. This reduces memory usage and eases the management of persistent data.

A servlet can be run by a servlet container in a restrictive environment, called a sandbox. This is similar to an applet that runs in the sandbox of the web browser. This enables restricted use of potentially harmful servlets. CGI programs can of course also sandbox themselves, since they are simply OS processes.

Technologies like FastCGI and its derivatives (including SCGI, AJP) do not exhibit the performance disadvantages of CGI, incurred by the constant process spawning. They are, however, roughly as simple as CGI. They are therefore also in contrast with servlets which are substantially more complex.

**1.2.7 XML**

In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specification]—all of them free open standards—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

The essence of why extensible markup languages are necessary is explained at Markup language (for example, see Markup language § XML) and at Standard Generalized Markup Language.

Hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, SVG, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also provided the base language for communication protocols such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration. Apple has an implementation of a registry based on XML.

XML has come into common use for the interchange of data over the Internet. IETF RFC:3023, now superseded by RFC:7303, gave rules for the construction of Internet Media Types for use when sending XML. It also defines the media types application/xml and text/xml, which say only that the data is in XML, and nothing about its semantics. The use of text/xml has been criticized as a potential source of encoding problems and it has been suggested that it should be deprecated.

RFC 7303 also recommends that XML-based languages be given media types ending in +xml; for example image/svg+xml for SVG.

Further guidelines for the use of XML in a networked context appear in RFC 3470, also known as IETF BCP 70, a document covering many aspects of designing and deploying an XML-based language.

**XML as Deployment Descriptor**

A deployment descriptor (DD) refers to a configuration file for an artifact that is deployed to some container/engine.

In the Java Platform, Enterprise Edition, a deployment descriptor describes how a component, module or application (such as a web application or enterprise application) should be deployed. It directs a deployment tool to deploy a module or application with specific container options, security settings and describes specific configuration requirements. XML is used for the syntax of these deployment descriptor files.

For web applications, the deployment descriptor must be called web.xml and must reside in the WEB-INF directory in the web application root. For Java EE applications, the deployment descriptor must be named application.xml and must be placed directly in the META-INF directory at the top level of the application .ear file.

In Java EE, there are two types of deployment descriptors: "Java EE deployment descriptors" and "runtime deployment descriptors". The Java EE deployment descriptors are defined by the language specification, whereas the runtime descriptors are defined by the vendor of each container implementation. For example, the web.xml file is a standard Java EE deployment descriptor, specified in the Java Servlet specification, but the sun-web.xml file contains configuration data specific to the Sun GlassFish Enterprise Server implementation.

Java web applications use a deployment descriptor file to determine how URLs map to servlets, which URLs require authentication, and other information. This file is named web.xml, and resides in the app's WAR under the WEB-INF/ directory. web.xml is part of the servlet standard for web applications.

A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests. When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.

The deployment descriptor is a file named web.xml. It resides in the app's WAR under the WEB-INF/ directory. The file is an XML file whose root element is <web-app>.

**1.2.8 JSP**

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified.

JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. This is a type of Model 2 architecture.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.



JSPs are usually used to deliver HTML and XML documents, but through the use of OutputStream, they can deliver other types of data as well.

The Web container creates JSP implicit objects like request, response, session, application, config, page, pageContext, out and exception. JSP Engine creates these objects during translation phase.

JSP pages use several delimiters for scripting functions. The most basic is <% ... %>, which encloses a JSP scriptlet. A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include <%= ... %> for expressions, where the scriptlet and delimiters are replaced with the result of evaluating the expression, and directives, denoted with <%@ ... %>.

Java code is not required to be complete or self-contained within a single scriptlet block. It can straddle markup content, provided that the page as a whole is syntactically correct. For example, any Java if/for/while blocks opened in one scriptlet must be correctly closed in a later scriptlet for the page to successfully compile.

Content that falls inside a split block of Java code (spanning multiple scriptlets) is subject to that code. Content inside an if block will only appear in the output when the if condition evaluates to true. Likewise, content inside a loop construct may appear multiple times in the output, depending upon how many times the loop body runs.

A JavaServer Pages compiler is a program that parses JSPs, and transforms them into executable Java Servlets. A program of this type is usually embedded into the application server and run automatically the first time a JSP is accessed, but pages may also be precompiled for better performance, or compiled as a part of the build process to test for errors.

Some JSP containers support configuring how often the container checks JSP file timestamps to see whether the page has changed. Typically, this timestamp would be set to a short interval (perhaps seconds) during software development, and a longer interval (perhaps minutes, or even never) for a deployed Web application.

In 2000, Jason Hunter, author of "Java Servlet Programming", criticized JSP for either tempting or requiring the programmer to mix Java code and HTML markup, although he acknowledged it would "wean people off" of Microsoft's Active Server Pages. Later, he added a note to his site saying that JSP had improved since 2000, but also cited its competitors, Apache Velocity and Tea (template language).

**1.2.9 Google Cloud DataStore**

Google Cloud Datastore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. Cloud Datastore features include:

* Atomic transactions. Cloud Datastore can execute a set of operations where either all succeed, or none occur.
* High availability of reads and writes. Cloud Datastore runs in Google data centers, which use redundancy to minimize impact from points of failure.
* Massive scalability with high performance. Cloud Datastore uses a distributed architecture to automatically manage scaling. Cloud Datastore uses a mix of indexes and query constraints so your queries scale with the size of your result set, not the size of your data set.
* Flexible storage and querying of data. Cloud Datastore maps naturally to object-oriented and scripting languages, and is exposed to applications through multiple clients. It also provides a SQL-like query language.
* Balance of strong and eventual consistency. Cloud Datastore ensures that entity lookups by key and ancestor queries always receive strongly consistent data. All other queries are eventually consistent. The consistency models allow your application to deliver a great user experience while handling large amounts of data and users.
* Encryption at rest. Cloud Datastore automatically encrypts all data before it is written to disk and automatically decrypts the data when read by an authorized user. For more information, see Server-Side Encryption.
* Fully managed with no planned downtime. Google handles the administration of the Cloud Datastore service so you can focus on your application. Your application can still use Cloud Datastore when the service receives a planned upgrade.

While the Cloud Datastore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects.



Unlike rows in a relational database table, Cloud Datastore entities of the same kind can have different properties, and different entities can have properties with the same name but different value types. These unique characteristics imply a different way of designing and managing data to take advantage of the ability to scale automatically. In particular, Cloud Datastore differs from a traditional relational database in the following important ways:

* Cloud Datastore is designed to automatically scale to very large data sets, allowing applications to maintain high performance as they receive more traffic:
* Cloud Datastore writes scale by automatically distributing data as necessary.
* Cloud Datastore reads scale because the only queries supported are those whose performance scales with the size of the result set (as opposed to the data set). This means that a query whose result set contains 100 entities performs the same whether it searches over a hundred entities or a million. This property is the key reason some types of queries are not supported.
* Because all queries are served by previously built indexes, the types of queries that can be executed are more restrictive than those allowed on a relational database with SQL. In particular, Cloud Datastore does not include support for join operations, inequality filtering on multiple properties, or filtering on data based on results of a subquery.
* Unlike traditional relational databases which enforce a schema, Cloud Datastore doesn't require entities of the same kind to have a consistent property set (although you can choose to enforce such a requirement in your own application code).

Cloud Datastore is ideal for applications that rely on highly available structured data at scale. You can use Cloud Datastore to store and query all of the following types of data:

* Product catalogs that provide real-time inventory and product details for a retailer.
* User profiles that deliver a customized experience based on the user’s past activities and preferences.
* Transactions based on ACID properties, for example, transferring funds from one bank account to another.

App Engine's Python standard runtime connects to Cloud Datastore using the NDB Client Library. The NDB Client Library provides persistent storage in a schemaless object datastore. It supports automatic caching, sophisticated queries, and atomic transactions.

You cannot use the Cloud Datastore client library with Python applications in the App Engine standard environment.

**1.3 Tools Used**

**1.3.1 Eclipse IDE**

****

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge.[8] The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation (Equinox) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles.

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License. It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

**History**

Eclipse was inspired by the Smalltalk-based VisualAge family of integrated development environment (IDE) products. Although fairly successful, a major drawback of the VisualAge products was that developed code was not in a component-based software engineering model. Instead, all code for a project was held in a compressed lump (somewhat like a zip file but in a proprietary format called .dat). Individual classes could not be easily accessed, certainly not outside the tool. A team primarily at the IBM Cary NC lab developed the new product as a Java-based replacement. In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open-source software. It is estimated that IBM had already invested nearly $40 million by that time. The original members were Borland, IBM, Merant, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, and WebGain. The number of stewards increased to over 80 by the end of 2003. In January 2004, the Eclipse Foundation was created.

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture.

The Association for Computing Machinery recognized Eclipse with the 2011 ACM Software Systems Award on 26 April 2012.

The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released. Some projects require dual licensing, for which the Eclipse Distribution License (EDL) is available, although use of this license must be applied for and is considered on a case-by-case basis.

Eclipse was originally released under the Common Public License, but was later re-licensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL).

According to Lee Nackman, Chief Technology Officer of IBM's Rational division (originating in 2003) at that time, the name "Eclipse" (dating from at least 2001) was not a wordplay on Sun Microsystems, as the product's primary competition at the time of naming was Microsoft Visual Studio, which Eclipse was to eclipse.

Different versions of Eclipse have been given different science-related names. The versions named after Callisto, Europa, and Ganymede, which are moons of Jupiter, were followed by a version named after Galileo the discoverer of those moons. These were followed by two sun-themed names, Helios of Greek mythology, and Indigo, one of the seven colors of a rainbow (which is produced by the sun). The version after that, Juno, has a triple meaning: a Roman mythological figure, an asteroid, and a spacecraft to Jupiter. Kepler, Luna, and Mars continued the astronomy theme, and then a new theme of chemical elements began, consisting of Neon and Oxygen. Photon represents a return to sun-themed names.

Eclipse uses plug-ins to provide all the functionality within and on top of the runtime system. Its runtime system is based on Equinox, an implementation of the OSGi core framework specification.

In addition to allowing the Eclipse Platform to be extended using other programming languages, such as C and Python, the plug-in framework allows the Eclipse Platform to work with typesetting languages like LaTeX and networking applications such as telnet and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with support for other version control systems provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. Thus, every plug-in developed integrates with Eclipse in the same way as other plug-ins; in this respect, all features are "created equal".[citation needed] Eclipse provides plug-ins for a wide variety of features, some of which are from third parties using both free and commercial models. Examples of plug-ins include for Unified Modeling Language (UML), for Sequence and other UML diagrams, a plug-in for DB Explorer, and many more.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in Java incremental compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace resource is refreshed afterward.

Eclipse implements the graphical control elements of the Java toolkit called Standard Widget Toolkit (SWT), whereas most Java applications use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate graphical user interface layer called JFace, which simplifies the construction of applications based on SWT. Eclipse was made to run on Wayland during a Google Summer of Code (GSoC) Project in 2014.

As of 2017, language packs being developed by the Babel Project provide translations into over 40 natural languages.

**Server Platform**

Eclipse supports development for Tomcat, GlassFish and many other servers and is often capable of installing the required server (for development) directly from the IDE. It supports remote debugging, allowing a user to watch variables and step through the code of an application that is running on the attached server.

**Web Tools Platform**

The Eclipse Web Tools Platform (WTP) project is an extension of the Eclipse platform with tools for developing Web and Java EE applications. It includes source and graphical editors for a variety of languages, wizards and built-in applications to simplify development, and tools and APIs to support deploying, running, and testing apps.

**Modelling Platform**

The Modeling project contains all the official projects of the Eclipse Foundation focusing on model-based development technologies. All are compatible with the Eclipse Modeling Framework created by IBM. Those projects are separated in several categories: Model Transformation, Model Development Tools, Concrete Syntax Development, Abstract Syntax Development, Technology and Research, and Amalgam.

Model Transformation projects uses Eclipse Modeling Framework (EMF) based models as an input and produce either a model or text as an output. Model to model transformation projects includes ATLAS Transformation Language (ATL), an open source transformation language and toolkit used to transform a given model or to generate a new model from a given EMF model. Model to text transformation projects contains Acceleo, an implementation of MOFM2T, a standard model to text language from the Object Management Group(OMG). The Acceleo code generator can generate any textual language (Java, PHP, Python, etc.) from EMF based models defined with any metamodel (Unified Modeling Language(UML), Systems Modeling Language (SysML), etc.). It is open-source.

Model Development Tools projects are implementations of various modeling standards used in the industry, and their toolkits. Among those projects can be found implementations of several standards:

* Unified Modeling Language (UML)
* Systems Modeling Language (SysML)
* Object Constraint Language (OCL)
* Business Process Model and Notation (BPMN)
* Interactive Media Manager (IMM)
* Semantics of Business Vocabulary and Business Rules (SBVR)
* XML Schema (XSD)
* National Electronic Distributors Association (NEDA)

The Concrete Syntax Development project contains the Graphical Modeling Framework, an Eclipse-based framework dedicated to the graphical representation of EMF based models.

The Abstract Syntax Development project hosts the Eclipse Modeling Framework, core of most of the modeling project of the Eclipse Foundation and the framework available for EMF like Connected Data Objects (CDO), EMF query or EMF validation.

Technology and Research projects are prototypes of Modeling project; this project is used to host all the modeling projects of the Eclipse Foundation during their incubation phase.

Amalgam provides the packaging and integration between all the available modeling tools for the Eclipse package dedicated to modeling tools.

**1.3.2 Google App Engine**

Google App Engine (often referred to as GAE or simply App Engine) is a web framework and cloud computing platform for developing and hosting web applications in Google-managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.



Google App Engine is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or instance hours required by the application. It was first released as a preview version in April 2008 and came out of preview in September 2011.

**Runtime and Framework**

Supported programming languages include Python, Ruby, Java (and, by extension, other JVM languages such as Kotlin, Groovy, JRuby, Scala, Clojure), Go, and PHP. Node.js is also available in the flexible environment. Google has said that it plans to support more languages in the future, and that the Google App Engine has been written to be language independent. C# is also supported. Arbitrary Docker containers are also supported.

Python web frameworks that run on Google App Engine include Django, CherryPy, Pyramid, Flask, web2py and webapp2, as well as a custom Google-written webapp framework and several others designed specifically for the platform that emerged since the release. Any Python framework that supports the WSGI using the CGI adapter can be used to create an application; the framework can be uploaded with the developed application. Third-party libraries written in pure Python may also be uploaded.

Google App Engine supports many Java standards and frameworks. Core to this is the servlet 2.5 technology using the open-source Jetty Web Server, along with accompanying technologies such as JSP. JavaServer Faces operates with some workarounds. A newer release of App Engine Standard Java in Beta supports Java8, Servlet 3.1 and Jetty9.

Though the integrated database, Google Cloud Datastore, may be unfamiliar to programmers, it is easily accessed and supported with JPA, JDO, and by the simple low-level API. There are several alternative libraries and frameworks you can use to model and map the data to the database such as Objectify, Slim3 and Jello framework.

The Spring Framework works with GAE. However, the Spring Security module (if used) requires workarounds. Apache Struts 1 is supported, and Struts 2 runs with workarounds.

The Django web framework and applications running on it can be used on App Engine with modification. Django-nonrel aims to allow Django to work with non-relational databases and the project includes support for App Engine.

**Reliability and Support**

All billed App Engine applications have a 99.95% uptime SLA.

App Engine is designed in such a way that it can sustain multiple datacenter outages without any downtime. This resilience to downtime is shown by the statistic that the High Replication Datastore saw 0% downtime over a period of a year.

Paid support from Google engineers is offered as part of Premier Accounts. Free support is offered in the App Engine Groups, Stack Overflow, Server Fault, and GitHub. However assistance by a Google staff member is not guaranteed.

**Bulk Downloading**

SDK version 1.2.2 adds support for bulk downloads of data using Python. The open source Python projects gaebar, approcket, and gawsh also allow users to download and back up App Engine data. No method for bulk downloading data from GAE using Java currently exists.

**Restrictions**

Developers have read-only access to the filesystem on App Engine. Applications can use only virtual filesystems, like gae-filestore.

App Engine can only execute code called from an HTTP request (scheduled background tasks allow for self calling HTTP requests).

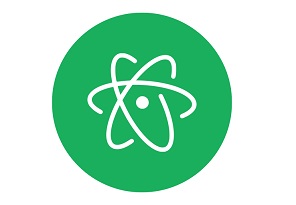
Users may upload arbitrary Python modules, but only if they are pure-Python; C and Pyrex modules are not supported.

Java applications may only use a subset (The JRE Class White List) of the classes from the JRE standard edition. This restriction does not exist with the App Engine Standard Java8 runtime.

A process started on the server to answer a request can't last more than 60 seconds (with the 1.4.0 release, this restriction does not apply to background jobs anymore).

Does not support sticky sessions (a.k.a. session affinity), only replicated sessions are supported including limitation of the amount of data being serialized and time for session serialization.

**1.3.3 Atom**

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in CoffeeScript and Less. It can also be used as an integrated development environment (IDE). Atom was released from beta, as version 1.0, on June 25, 2015. Its developers call it a "hackable text editor for the 21st Century".

**Language Support**

Using the default plugins, the following languages are supported in some aspect as of v1.5.1:

C/C++, C#, Clojure, CSS, CoffeeScript, GitHub Flavored Markdown, Go, Git, HTML, JavaScript, Java, JSON, Julia, Less, Make, Mustache, Objective-C, PHP, Perl, Property List (Apple), Python, Ruby on Rails, Ruby, Sass, Shell script, Scala, SQL, TOML, XML, YAML.

**License**

Initially, extension packages for Atom and anything not part of Atom's core were released under an open-source license. On 6 May 2014, the rest of Atom, including the core application, its package manager, as well as its desktop framework Electron, were released as free and open-source software under the MIT License.

**Privacy**

There was initially concern and discussion about two opt-out packages that report various data to external servers. However, those packages are now opt-in with a verbose dialog at the first initial launch:

* Metrics package: Reports usage information to Google Analytics. Reports to Google Analytics, including a unique UUID v4 random identifier. According to the authors, this is to determine the performance and know the most-used functions. This feature can be disabled by the user by opening the Settings View, searching for the metrics package, and disabling it.
* Exception-reporting package: Reports uncaught Atom exceptions to bugsnag.com.

**1.3.4 Amazon Web Services**



Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis with a free-tier option available for 12 months. The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the internet. AWS's version of virtual computers have most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for their own and their customers' benefit.

The AWS technology is implemented at server farms throughout the world, and maintained by the Amazon subsidiary. Fees are based on a combination of usage, the hardware/OS/software/networking features chosen by the subscriber, required availability, redundancy, security, and service options. Based on what the subscriber needs and pays for, they can reserve a single virtual AWS computer, a cluster of virtual computers, a physical (real) computer dedicated for their exclusive use, or even a cluster of dedicated physical computers. As part of the subscription agreement, Amazon manages, upgrades, and provides industry-standard security to each subscriber's system. AWS operates from many global geographical regions including 6 in North America.

In 2016, AWS comprised more than 70 services spanning a wide range including computing, storage, networking, database, analytics, application services, deployment, management, mobile, developer tools, and tools for the Internet of Things. The most popular include Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Most services are not exposed directly to end users, but instead offer functionality through APIs for developers to use in their applications. Amazon Web Services’ offerings are accessed over HTTP, using the REST architectural style and SOAP protocol.

Amazon markets AWS to subscribers as a way of obtaining large scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways.

**Availability and Topology**

AWS has announced that 3 new regions (having 7 Availability Zones) will come online throughout 2017 in China, India, and the United Kingdom.

Each region is wholly contained within a single country and all of its data and services stay within the designated region.[3] Each region has multiple "Availability Zones", which consist of one or more discrete data centers, each with redundant power, networking and connectivity, housed in separate facilities. Availability Zones do not automatically provide additional scalability or redundancy within a region, since they are intentionally isolated from each other to prevent outages from spreading between Zones. Several services can operate across Availability Zones (e.g., S3, DynamoDB) while others can be configured to replicate across Zones to spread demand and avoid downtime from failures.

As of December 2014, Amazon Web Services operated an estimated 1.4 Million servers across 28 availability zones. The global network of AWS Edge locations consists of 54 points of presence worldwide, including locations in the United States, Europe, Asia, Australia, and South America.

In 2014, AWS committed to achieving 100% renewable energy usage. In the United States, AWS's partnerships with renewable energy providers include:

* Community Energy of Virginia, a solar farm coming online in 2016, to support the US East region.
* Pattern Development, in January 2015, to construct and operate Amazon Wind Farm Fowler Ridge.
* Iberdrola Renewables, LLC, in July 2015, to construct and operate Amazon Wind Farm US East.
* EDP Renewables North America, in November 2015, to construct and operate Amazon Wind Farm US Central.
* Tesla Motors, to apply battery storage technology to address power needs in the US West (Northern California) region.

**History**

The AWS platform was launched in July 2002 to "expose technology and product data from Amazon and its affiliates, enabling developers to build innovative and entrepreneurial applications on their own." In the beginning, the platform consisted of only a few disparate tools and services. Then in late 2003, the AWS concept was publicly reformulated when Chris Pinkham and Benjamin Black presented a paper describing a vision for Amazon's retail computing infrastructure that was completely standardized, completely automated, and would rely extensively on web services for services such as storage and would draw on internal work already underway. Near the end of their paper, they mentioned the possibility of selling access to virtual servers as a service, proposing the company could generate revenue from the new infrastructure investment. In November 2004, the first AWS service launched for public usage: Simple Queue Service (SQS). Thereafter Pinkham and lead developer Christoper Brown developed the Amazon EC2 service, with a team in Cape Town, South Africa.

Amazon Web Services was officially re-launched on March 14, 2006, combining the three initial service offerings of Amazon S3 cloud storage, SQS, and EC2. The AWS platform finally provided an integrated suite of core online services, as Chris Pinkham and Benjamin Black had proposed back in 2003, as a service offered to other developers, web sites, client-side applications, and companies. Andy Jassy, AWS founder and vice president in 2006, said at the time that Amazon S3 (one of the first and most scalable elements of AWS) "helps free developers from worrying about where they are going to store data, whether it will be safe and secure, if it will be available when they need it, the costs associated with server maintenance, or whether they have enough storage available. Amazon S3 enables developers to focus on innovating with data, rather than figuring out how to store it." His quote marks a milestone in the Internet's history, when massive managed resources became available to developers worldwide, allowing them to offer new scalable web-enabled technologies. In 2016 Jassy was promoted to CEO of the division. Reflecting the success of AWS, his annual compensation in 2017 hit nearly $36 million.

To support industry-wide training and skills standardization, AWS began offering a certification program for computer engineers, on April 30, 2013, to highlight expertise in cloud computing.

James Hamilton, an AWS engineer, wrote a retrospective article in 2016 to highlight the ten-year history of the online service from 2006 to 2016. As an early fan and outspoken proponent of the technology, he had joined the AWS engineering team in 2008.

**Growth and Profitability**

In November 2010, it was reported that all of Amazon.com's retail sites had been completely moved under the AWS umbrella. Prior to 2012, AWS was considered a part of Amazon.com and so its revenue was not delineated in Amazon financial statements. In that year industry watchers for the first time estimated AWS revenue to be over $1.5 billion.

In April 2015, Amazon.com reported AWS was profitable, with sales of $1.57 billion in the first quarter of the year and $265 million of operating income. Founder Jeff Bezos described it as a fast-growing $5 billion business; analysts described it as "surprisingly more profitable than forecast". In October 2015, Amazon.com said in its Q3 earnings report that AWS's operating income was $521 million, with operating margins at 25 percent. AWS's 2015 Q3 revenue was $2.1 billion, a 78% increase from 2014's Q3 revenue of $1.17 billion. 2015 Q4 revenue for the AWS segment increased 69.5% y/y to $2.4 billion with 28.5% operating margin, giving AWS a $9.6 billion run rate. In 2015, Gartner estimated that AWS customers are deploying 10x more infrastructure on AWS than the combined adoption of the next 14 providers.

In 2016 Q1, revenue was $2.57 billion with net income of $604 million, a 64% increase over 2015 Q1 that resulted in AWS being more profitable than Amazon's North American retail business for the first time. In the first quarter of 2016, Amazon experienced a 42% rise in stock value as a result of increased earnings, of which AWS contributed 56% to corporate profits.

With a 50% increase in revenues the past few years, AWS is predicted to have $13 billion in revenue in 2017.

**Customer base**

AWS adoption has increased since launch in 2002.

On March 14, 2006, Amazon said in a press release: "More than 150,000 developers have signed up to use Amazon Web Services since its inception."

In June 2007, Amazon claimed that more than 180,000 developers had signed up to use Amazon Web Services.

In November 2012, AWS hosted its first customer event in Las Vegas.

On May 13, 2013, AWS was awarded an Agency Authority to Operate (ATO) from the U.S. Department of Health and Human Services under the Federal Risk and Authorization Management Program.

In October 2013, it was revealed that AWS was awarded a $600M contract with the CIA.

During August 2014, AWS received Department of Defense-Wide provisional authorization for all U.S. Regions.

During the 2015 re:Invent keynote, AWS disclosed that they have more than a million active customers every month in 190 countries, including nearly 2,000 government agencies, 5,000 education institutions and more than 17,500 nonprofits.

On April 5 2017, AWS and DXC Technology (formed from a merger of CSC and HPE) announced an expanded alliance to increase access of AWS features for enterprise clients in existing data centers.

Notable customers include NASA, the Obama presidential campaign of 2012, Kempinski Hotels, and Netflix.

**Chapter 2**

**2.1 Introduction**

This chapter gives an overview about the aim, objectives, background and operation environment of the system.

**2.2 Project Aim and Objectives**

The project aim and objectives will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

* Avoid wastage of papers
* Avoid wastage of resources
* Simplified User Experience
* Can contact easily with the seller
* Easy to upload products
* User Friendly Interface
* An always available web-application

This web-application basic objective is to avoid wastage of papers and simplified hands on experience to the user. Most of the people in the rural areas do not have availability of laptops and computers, but recent survey have shown that number of computer and mobile users are increasing drastically in the rural areas and hence many people are being connected through computers and mobiles. This web based application provide its service to everyone who are in GGSIPU colleges and want notes, reference books and drafters etc. As the number of computer and mobile users are increasing, this web-application can reach a lot of people.

As you can see in your college’s photocopy shop paper is wasted like water, by giving same notes every year. Any user who is registered in this website can upload their used resources like drafter, photocopied notes, handmade notes and reference books etc and junior students of same course can see these product in this website then these students can contact you for buying these items if they need. Senior students can also sell their resources to junior students for free as per their desired wish.

**2.3 Operational Environment**

**2.3.1 Minimum Requirements**

**-** Windows 7(or later) or Linux or MacOS or Mobile OS

- Internet Explorer 7 or Mobile Default Browser

**2.3.2 Recommended Requirement**

**-** Windows 7(or later) or Linux or MacOS or Mobile OS

- Google Chrome or Safari or Mozilla Firefox

- Cookies Enabled

**2.4 System Aim**

This web-application has been aiming at the latest features of the web-application platform. Since this major update of the web-application system had brought a lot of changes in the application presentation and User Interface Features, this web-application will give its best performance in browser like Google Chrome, Firefox and Safari.

Backward compatibility have been considered while building this application and this will run on previous versions of browser too but may not be able to show some of the animations and related effects.

**2.5 System Design**

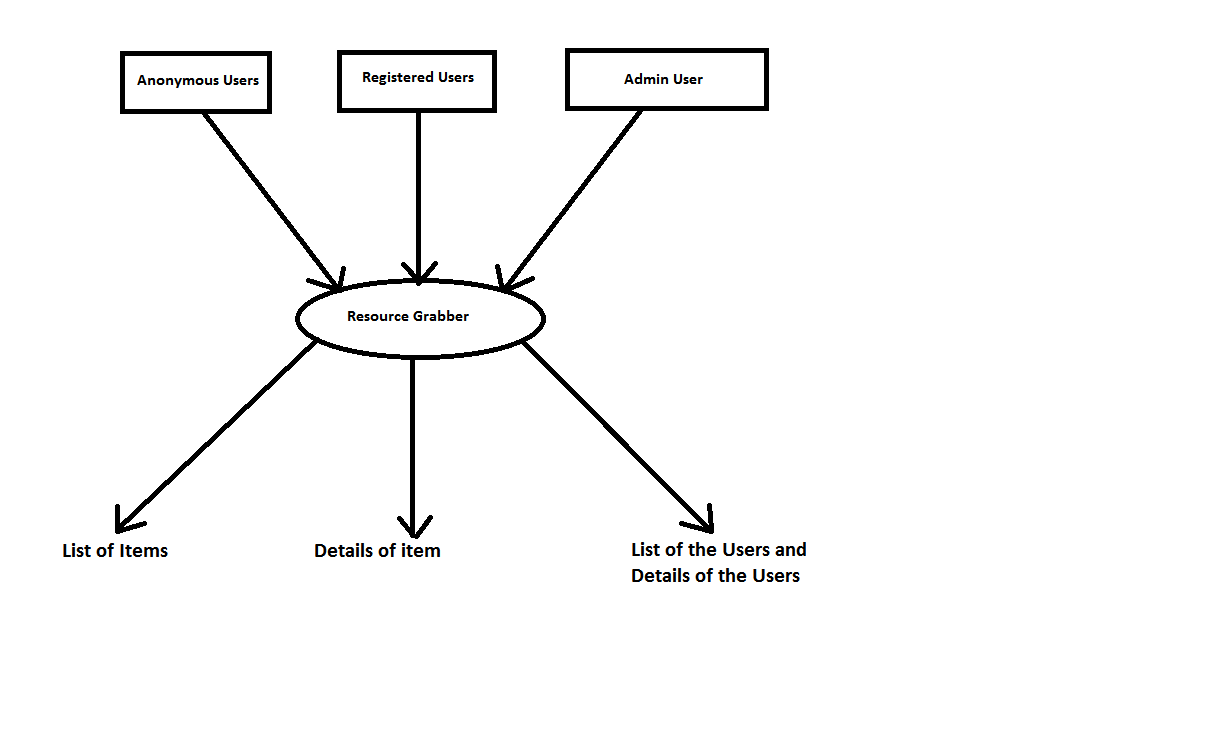
**2.5.1 Table Design**

This web-application uses Google Cloud Datastore which are accessed by the application during the processing period.

The Datastore is stores in the Google Cloud Platform.

This application consist of a single Datastore consisting of two entities which shows the user details and product/item details.

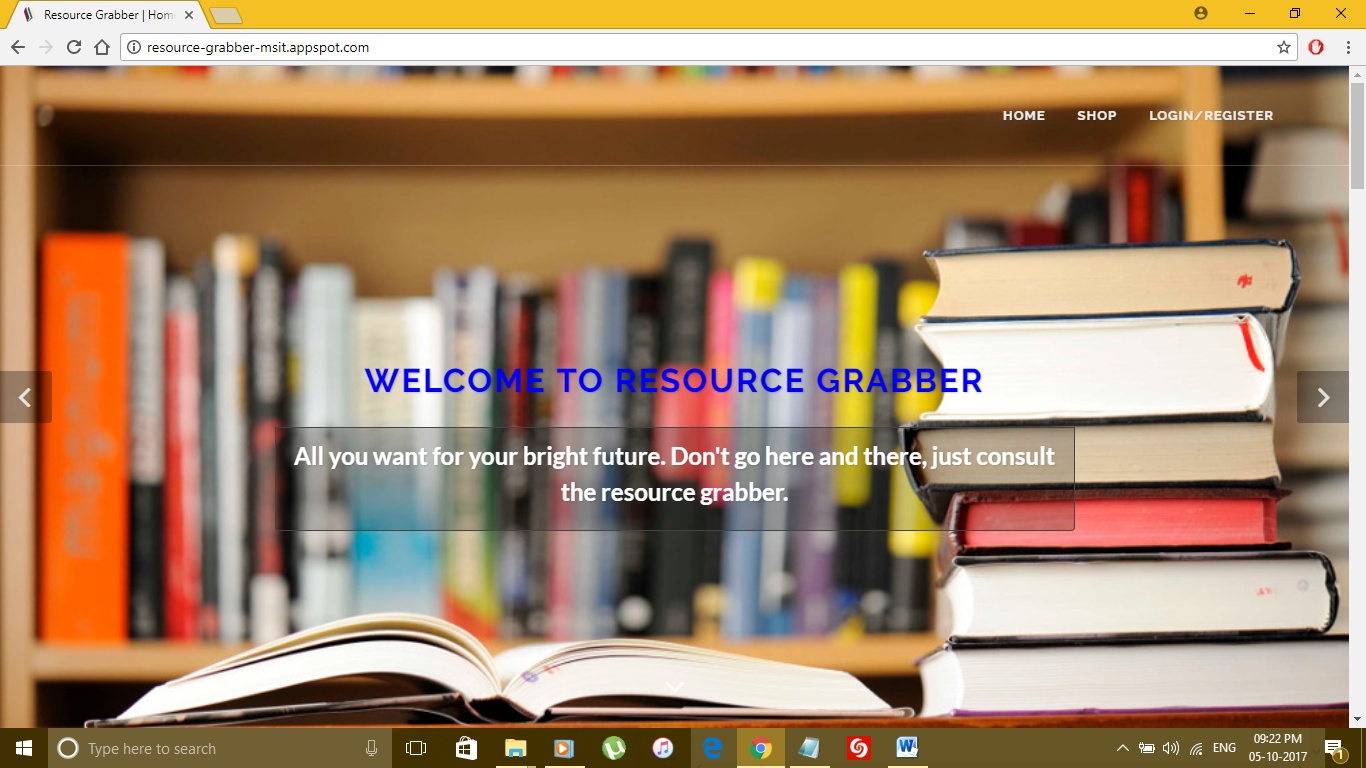
**2.6 Data flow Diagram**



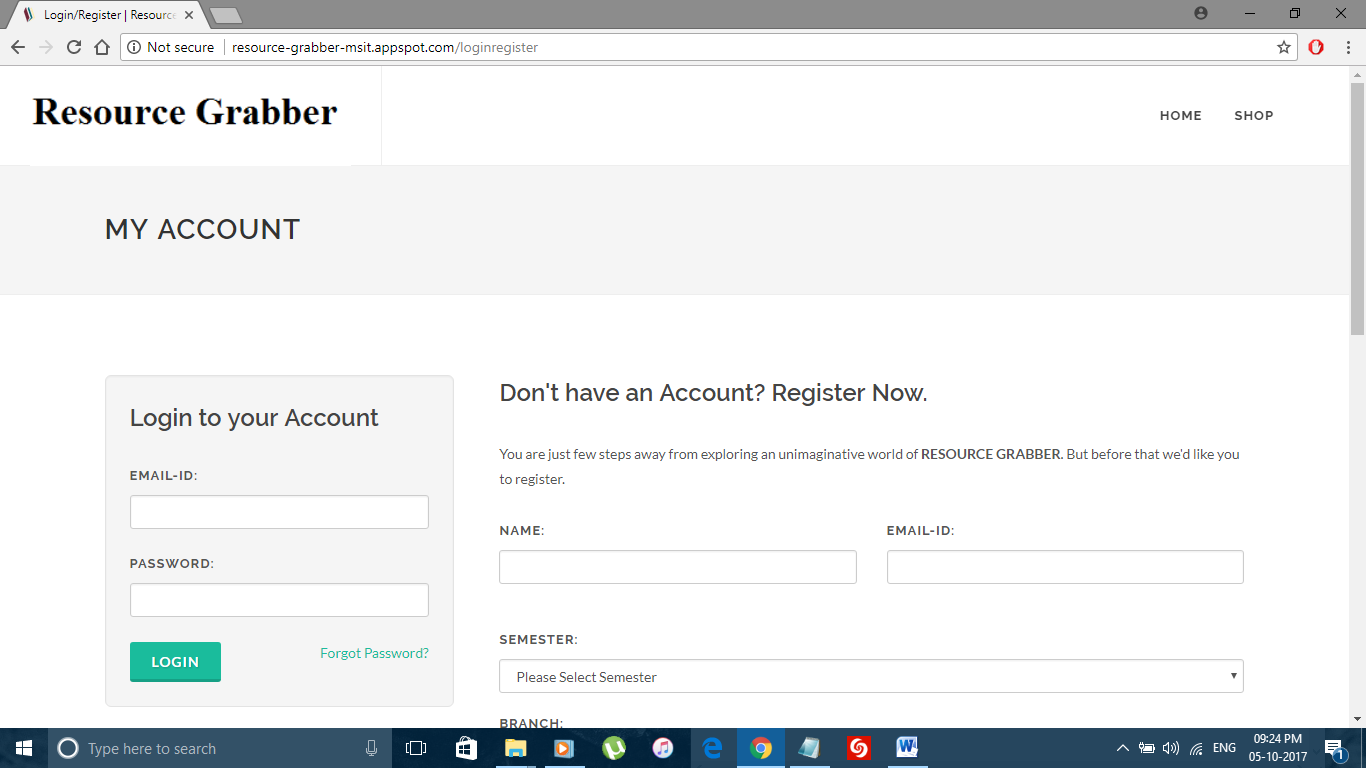
**2.8 Web-Application Implementation**

**2.8.1 Screenshots**

**Home Page**

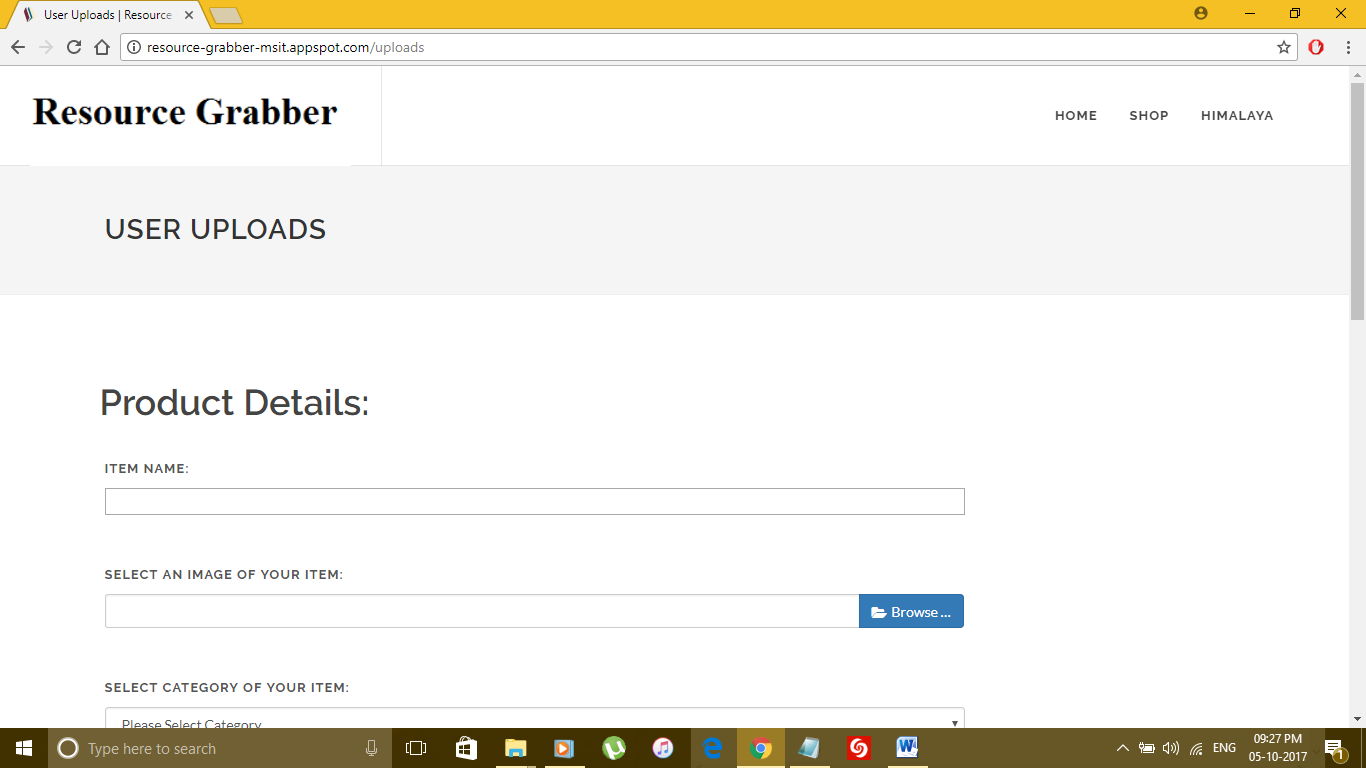


**Login/Register Page**



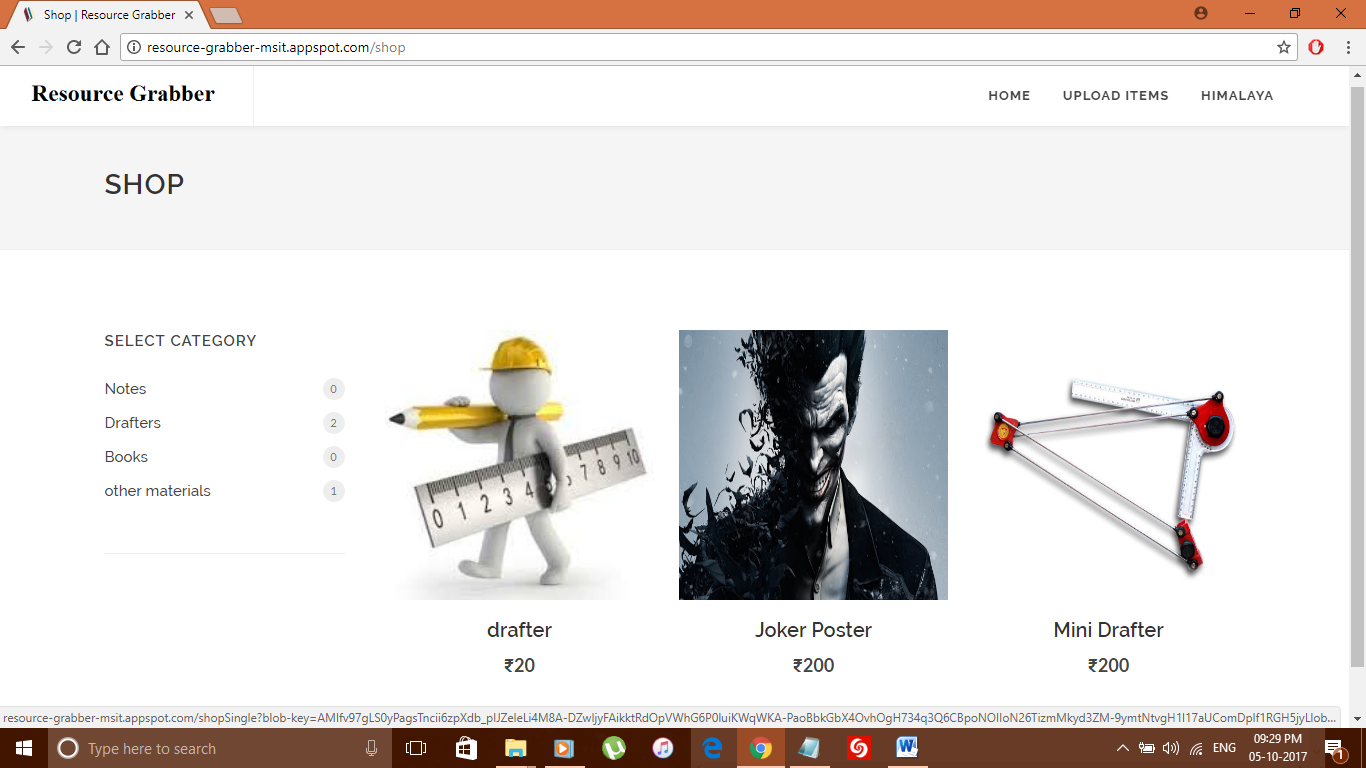
This Module provides an area for Login and Registration.

**User Uploads**



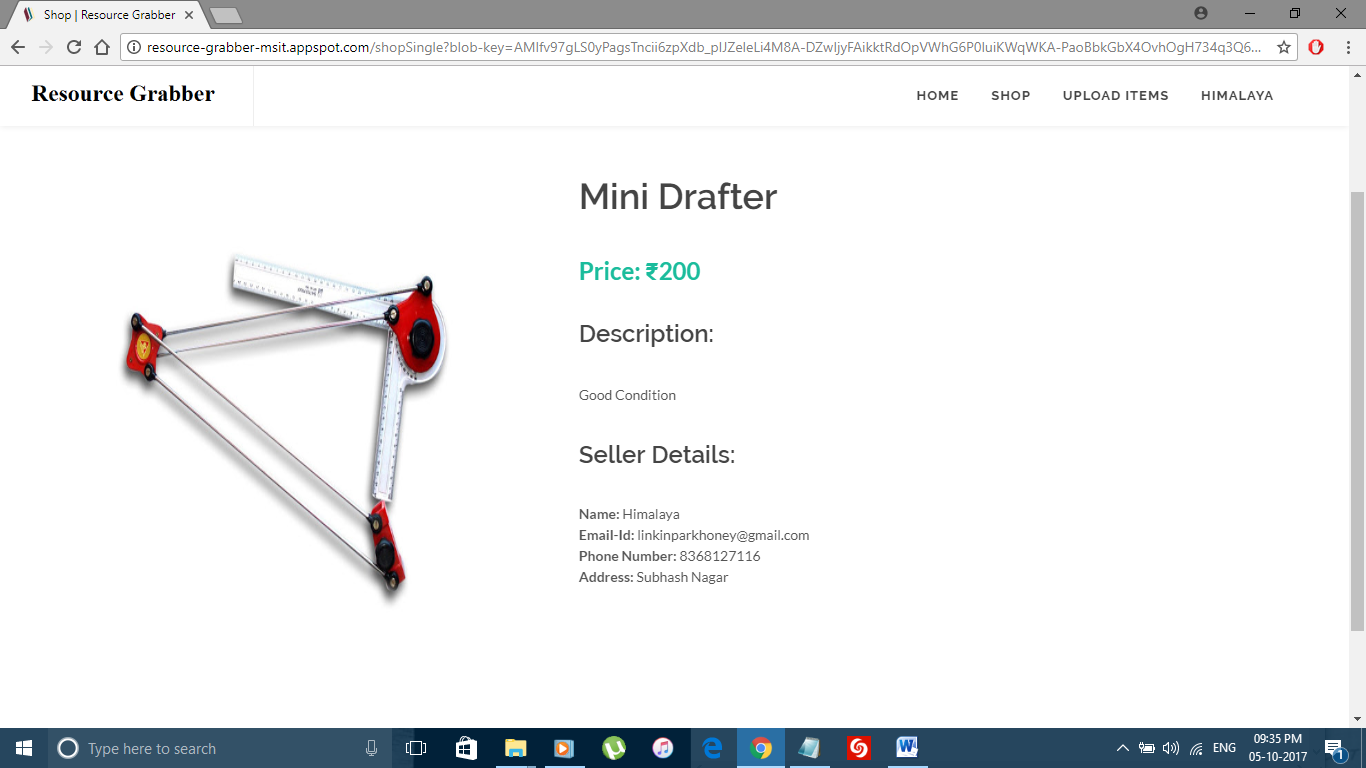
This module provides an area where registered user can upload their items/products.

**Shop Page**



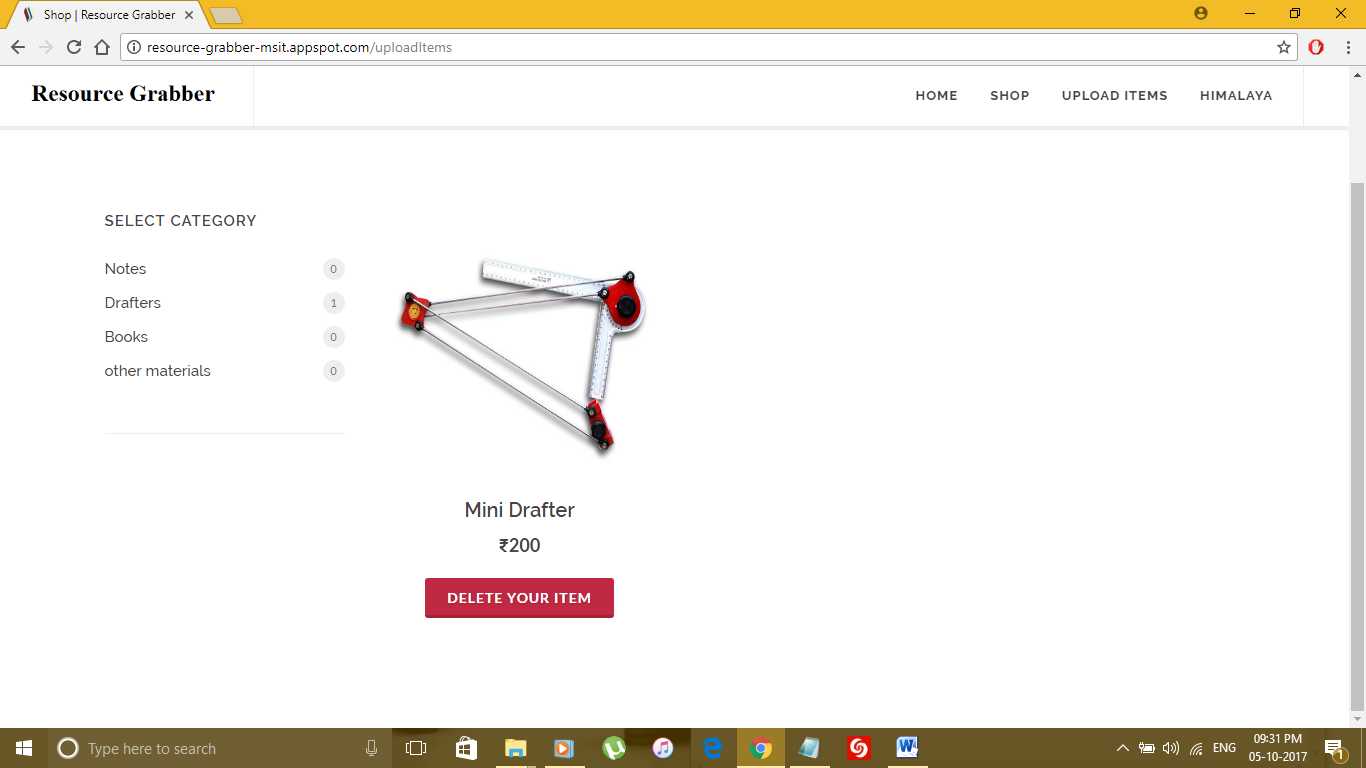
This page provides an area where anonymous and registered user can see the products.

**Product Description**



This module provides an area where users can see the details of the items and seller details.

**Uploaded Items Page**



This module provides an area where registered users can manage their items.

**Bibliography**

1. <https://en.wikipedia.org>

2. <https://cloud.google.com>

3. <https://www.javatpoint.com>

4. <https://www.tutorialspoint.com>

5. <https://www.w3schools.com>