# 

# **AutoSeats**

Table of Contents

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Introduction | | |  | 1 |
| 2 | **Literature Review** | | |  | **2** |
| **3** | **System Analysis** | | |  | **3** |
|  | 3.1 | Preface | |  | 3 |
| 3.2 | Introducion | |  | 3 |
| 3.3 | Glossary | |  | 3 |
| 3.4 | User Requirements Definition | |  | 4 |
| 3.5 | The products and process features | |  | 4 |
| 3.6 | Modules | |  | 5 |
| 3.6.1 | Exam Cell Module |  | 5 |
| 3.6.2 | Student Module |  | 6 |
| 3.7 | SDLC | |  | 6 |
| 3.8 | Technologys tack | |  | 11 |
| 3.8.1 | HTML |  | 11 |
| 2.8.2 | CSS |  | 20 |
| 3.8.3 | Bootstrap |  | 24 |
| 3.8.4 | JavaScript |  | 27 |
| 3.8.5 | AJAX |  | 30 |
| 3.8.6 | ASP.NET |  | 32 |
| 3.8.7 | Microsoft SQL Server |  | 35 |
| 3.8.8 | C# |  | 38 |
| **4** | **Testing And Analysis** | | |  | **43** |
| 4.1 | Feasibility Study | |  | 43 |
| 4.1.1 | Economic Feasibility |  | 43 |
| 4.1.2 | Technical Feasibility |  | 43 |
| 4.1.3 | Behavioural Feasibility |  | 43 |
| 4.2 | Data Flow Diagram(DFD) | |  | 44 |
| 4.3 | ER Diagram | |  | 45 |
| 4.4 | Use Case | |  | 46 |
| 4.5 | Testing | |  | 48 |
| 4.6 | Test Execution | |  | 50 |
| **5** | **Result** | | |  | **52** |
| 5.1 | Conclusion | |  | 52 |
| 5.2 | Future Scope | |  | 52 |
| **6** | **Output** | | |  | **53** |
|  | *Bibliography* | | |  | 58 |

# **1. Introduction**

In present time, Universities are depend on their staff to make sitting plan of the students for examination. It is very difficult and time consuming. There is a constraint that to students, whose are appearing for same exam, can’t sit together. It is the tough task to manage by human resources. So there is very less randomness on sitting plan. It prepare once and use multiple time. Another problem is that it also difficult to find by students their seats.

AutoSeats is a web application software which does automated this process. We just need to one click to generate sitting plan for one exam and it will give you result. Then it send room number and seat number to students. You can also see the graphical view of class visiting web portal. It will be very helpful for universities to improve the efficiency of their exam.

More specifically, AutoSeats is a seats allocation system for universities to manage theirs students seating in exams. This software is designed to reduce the human power and increase the speed of seating arrangement process. Proposed system will be overcome deficiencies of existing manual system. It ensure that students of similar subjects could not sit together. This software also have a web portal which simulate classrooms. It will help students to find their seats virtually. Another feature is that it sends seat number and room number to each students. It also sends room no to invigilators. This system contains a relation database, web portal for each user.

**2. LITERATURE REVIEW**

In present time, Universities are totally depend on their staff to arrange the sitting plan. As the number of students increases complexity of sitting plan also increases. So faces a lot of problem on generation of sitting plan. It also take more time and extra effort. So if we can automate this thing then college staff will be in relief and process can be fast.

There are some software and algorithms which allocate sitting plan for competitive exam but they are not useful in universities regular examination. These software only generate the sitting plan and provides printable files but they don’t provide students or candidate to a portal to check their sitting plan online. So they must go to college then find on information board. Which also very hard in case of large universities.

An algorithm was published on IEEE to allocate the sitting plan but that was for basically competitive exams. Inn universities exam there are a lot of constraints which must be keep on mind in allocation of seats to students. So to make the all process easy, fast and easy accessible we need a software so that burden from stuff can be reduce and sitting arrangement will be more efficient.

1. **SYSTEM ANALYSIS**

**3.1 Preface**

This document has been written to apply a new version of SRS Software Requirements Specification depends on IEEE-STD-830-1998 standard. So, you must compare this document with this standard.

This is the first version for ***AutoSeats***

This document is the basic intended for any individual user, developer, tester, project manager or documentation writer that needs to understand the basic system architecture and its specifications.

**3.2 Introduction**

The purpose of this SRS document is to write the functional and nonfunctional user or system requirements that represent the characteristics of AutoSeats.

* AutoSeats is designed to provide a web portal to students to check their sitting plan and a portal to exam cell for generation and management of sitting plan.
* Provides a web portal for student to check their sitting plan.
* Exam Cell can generate, view and edit the sitting plan.
* Exam cell can add the room.
* Both can view the virtual room in portal.
  1. **Glossary**

|  |  |  |
| --- | --- | --- |
|  | Short name | Description |
| 1 | Server | System which have control on other system |
| 2 | Students/Exam Cell | Clients(End Users) |
| 3 | Administrator/Instructor | Who will watch Activity log file of others |
| 4 | Faculty member | A teacher in the faculty |

* 1. **User Requirements Definition**

The user requirement for this system is to make the system fast, flexible, less prone to error, reduce expenses and save the time.

* Exam Cell can add room a new.
* Exam can set availability of a room for sitting.
* Exam Cell can generate the sitting plan.
* After generation they can save or save after review.
* They can view the sitting plan of a specific day.
* They can view any room in virtual view
* Student can login and then check his sitting plan
* Student can also see virtual room to get specific position of his seat in room
  1. **The products and process features**
* This application provides a single click generation of sitting plan after selection of exam date and shift of exam.
* This application provides a simple and easy interface to upload time table.
* This application provides a well-designed GUI for addition of room.
* This application has virtual room view
* This application can send SMS and mail after allocation of sitting plan.
* Students can get their seat number and also can get specific position of that seat using virtual room view.
  1. **Modules**

This section gives overview of modules of the project. AutoSeats divided into two module on basis of end user types which are following

1. Exam Cell Module
2. Student Module
   * 1. **Exam Cell Module**

This Module consist the all functionality of exam cell department. Exam cell can do various task. Accessibility of this module features is limited to specific users. So authorization and verification part is also added in this module. Exam Cell modules provides functions like adding rooms, uploading time table, generation of sitting plan, view generated sitting plan, virtual room view for more realistic view.

Figure 3.1: Exam Cell Module Functions

* + 1. **Student Module**

This Module consist the all functionality for students. This modules provides students to their allotted seat with room no and students can further explore their seat in virtual room view.

Figure 3.2: Student Module Functions

* 1. **Software Development Life Cycle (SDLC) Model**

Software development life cycle (SDLC) is a series of phases that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs. The good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements.

Therefore, it is required to choose the right SDLC model according to the specific concerns and requirements of the project to ensure its success. There are various SDLC models which are following:

* Waterfall Model
* V-Shaped Model
* Evolutionary Prototyping Model
* Spiral Method (SDM)
* Iterative and Incremental Method
* Agile development

***Waterfall Model***

The Waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach and most widely known that was used for software development.

***V-Shaped Model***

It is an extension of the waterfall model, Instead of moving down in a linear way, the process steps are bent upwards after the implementation and coding phase, to form the typical V shape. The major difference between V-shaped model and waterfall model is the early test planning in the V-shaped model.

***Evolutionary Prototyping Model***

It refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed. It is an activity that can occur in software development. It used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team. This also will reduce the iterations may occur in waterfall approach and hard to be implemented due to the inflexibility of the waterfall approach. So, when the final prototype is developed, the requirement is considered to be frozen.

***Spiral Method (SDM)***

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.

***Iterative and Incremental Method***

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system.

***Agile development***

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams. It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, it can be used when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough.

*I used Iterative and Incremental model in development of AutoSeats*

**Iterative and Incremental Model**

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

It is used in shrink-wrap application and large system which built-in small phases or segments. Also, can be used in a system has separated components, for example, ERP system. Which we can start with the budget module as a first iteration and then we can start with inventory module and so forth.

***Iterative Model - Design***

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model –



Figure 3.3: Iterative Model

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

***Iterative Model - Application***

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios

* Requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
* There is a time to the market constraint.
* A new technology is being used and is being learnt by the development team while working on the project.
* Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
* There are some high-risk features and goals which may change in the future.

***Advantages***

* Produces business value early in the development lifecycle.
* Better use of scarce resources through proper increment definition.
* Can accommodate some change requests between increments.
* More focused on customer value than the linear approaches.
* Problems can be detected earlier.

***Disadvantages***

* Requires heavy documentation.
* Follows a defined set of processes.
* Defines increments based on function and feature dependencies.
* Requires more customer involvement than the linear approaches.
* Partitioning the functions and features might be problematic.
* Integration between iteration can be an issue if this is not considered during the development.
  1. **Technology Stack**

Auto Seats is web based application so we use many technologies to implement it. We make our website responsive and provide a good user interface.

***Front End Technologies***

* HTML5
* CSS3
* Bootstrap
* JavaScript
* Ajax
* ASP.NET

***Back End Technologies***

* Microsoft SQL Server 2017
* C# 7.0

WE use visual studio 2017 Community edition as IDE for development and developer edition of the Microsoft SQL Server 2017

* + 1. **HTML (Hyper Text Markup Language)**

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript which affect the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

***History***

In 1980, physicist Tim Berners-Lee, a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes from 1990 he listed "some of the many areas in which hypertext is used" and put an encyclopedia first.

The idea behind HTML was a modest one. When Tim Berners-Lee was putting together his first elementary browsing and authoring system for the Web, he created a quick little hypertext language that would serve his purposes. He imagined dozens, or even hundreds, of hypertext formats in the future, and smart clients that could easily negotiate and translate documents from servers across the Net. It would be a system similar to Claris XTND on the Macintosh, but would work on any platform and browser.

The problem, however, turned out to be in the simplicity of Berners-Lee's language. Since it was text-based, you could use any editor or word processor to create or convert documents for the Web. And there was just a handful of tags – anyone could master HTML in an afternoon. The Web flourished. Everyone started publishing. The rest is history.

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Tim Berners-Lee in late 1991. It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house Standard Generalized Markup Language (SGML)-based documentation format at CERN. Eleven of these elements still exist in HTML 4.

HTML is a markup language that web browsers use to interpret and compose text, images, and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS. Many of the text elements are found in the 1988 ISO technical report TR 9537 Techniques for using SGML, which in turn covers the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and markup; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the Internet Engineering Task Force (IETF) with the mid-1993 publication of the first proposal for an HTML specification, the "Hypertext Markup Language (HTML)" Internet Draft by Berners-Lee and Dan Connolly, which included an SGML Document type definition to define the grammar. The draft expired after six months, but was notable for its acknowledgment of the NCSA Mosaic browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes. Similarly, Dave Raggett's competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based.

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the World Wide Web Consortium (W3C). However, in 2000, HTML also became an international standard (ISO/IEC 15445:2000). HTML 4.01 was published in late 1999, with further errata published through 2001. In 2004, development began on HTML5 in the Web Hypertext Application Technology Working Group (WHATWG), which became a joint deliverable with the W3C in 2008, and completed and standardized on 28 October 2014.

***HTML Version***



Figure 3.4: HTML Version

**HTML 1.0**

HTML 1.0 was the first release of HTML to the world. Not many people were involved in website creation at the time, and the language was very limiting. There really wasn’t much you could do with it bar getting some simple text onto the web. But then, just that got the beardos a-foamin’ back in the day.

**HTML 2.0**

HTML 2.0 included everything from the original 1.0 specifications but added a few new features to the mix. [» HTML 2.0](http://www.w3.org/MarkUp/html-spec/) was the standard for website design until January 1997 and defined many core HTML features for the first time.

**HTML 3.0**

More and more people were getting into the HTML game around now, and while the previous standards offered some decent abilities to webmasters (as they became known), they thirsted for more abilities and tags. They wanted to enhance the look of their sites.

This is where trouble started. A company called Netscape was the clear leader in the browser market at the time, with a browser called Netscape Navigator. To appease the cries of the HTML authors, they introduced new *proprietary tags* and attributes into their Netscape Navigator browser. These new abilities were called **Netscape extension tags**. This caused big problems as other browsers tried to replicate the effects of these tags so as not to be left behind but could not get their browsers to display things the same way. This meant that if you designed a page with Netscape ETs, the page would look bad in other browsers. This caused confusion and irritation for the markup pioneers.

At this time, a HTML working group, led by a man named [» Dave Raggett](http://www.w3.org/People/Raggett/) introduced a new HTML draft, HTML 3.0. It included many new and improved abilities for HTML, and promised far more powerful opportunities for webmasters to design their pages. Sadly, the browsers were awfully slow in implementing any of the new improvements, only adding in a few and leaving out the rest. Partly, this failure can be attributed to the size of the overhaul; and so the HTML 3.0 spec was abandoned.

Thankfully, the people in charge noted this and so future improvements were always designed to be **modular**. This meant they could be added in stages, which makes it easier on the browser companies.

**HTML 3.2**

The browser-specific tags kept coming, and it became increasingly apparent that a standard needed to be found. To this end, the [» World Wide Web Consortium](http://www.w3.org/) (abbreviated to the *W3C*) was founded in 1994 to standardise the language and keep it evolving in the right direction. Their first work was code-named *WILBUR*, and later became known as [» HTML 3.2](http://www.w3.org/TR/REC-html32). This was a toned-down change to the existing standards, leaving many of the big steps forward for later versions. Most of the extensions tags that had been introduced by Netscape (and to a lesser-extent, Microsoft) did not make it into these new standards. It soon caught on and became the official standard in January ’97, and today practically all browsers support it fully.

**HTML 4.01**

HTML 4.0 was a large evolution of the HTML standards, and the last iteration of classic HTML. Early in development it had the code-name *COUGAR*. Most of the new functionality brought in this time is from the ill-fated HTML 3.0 spec, as well as a host of trimmings on old tags, a focus on internationalisation, and support for HTML’s new supporting presentational language, [cascading stylesheets](http://www.yourhtmlsource.com/stylesheets/).

HTML 4.0 was recommended by the W3C in December ’97 and became the official standard in April 1998. Browser support was undertaken surprisingly earnestly by Microsoft in their [Internet Explorer](http://www.yourhtmlsource.com/starthere/browserreview.html#ie6) browser, and the market-leading IE5 (and current successor IE6) have excellent support for almost all of the new tags and attributes. In comparison, Netscape’s terribly flawed Navigator 4.7 was inept when it came to HTML 4.0 and even basic CSS. Modern browsers however, are a vast improvement.

Once HTML 4.0 had been out for a little while, the documentation was revised and corrected in a few minor ways and was entitled HTML 4.01; the final version of the specification.

Head on over to the W3C site for the [» official documentation](http://www.w3.org/TR/html4/); and to read more about the new tags, attributes and redundancies brought about by this new standard, read our article, [HTML 4 Explained](http://www.yourhtmlsource.com/accessibility/html4explained.html).

**XHTML 1.0**

Close to the beginning of the 21st century the W3C issued their [» specifications](http://www.w3.org/TR/xhtml1/) of XHTML1.0 as a *recommendation*. Since January 26, 2000 it stands as the joint-standard with HTML 4.01. XHTML marks a departure from the way new specs have worked — it is an entirely new branch of HTML, incorporating the rigours of [» XML](http://www.w3.org/TR/REC-xml/), so that code must be properly written if it is to work once it reaches the reader’s browser. There weren’t many new or deprecated tags and attributes in XHTML, but some things changed with a view of increased accessibility and functionality. It’s mainly just a new set of coding rules. Read all about it properly in [XHTML Explained](http://www.yourhtmlsource.com/accessibility/xhtmlexplained.html).

**HTML5**

After HTML 4.01 and XHTML 1.0, the guys who were in control of HTML’s direction got sidetracked working on a new proposal for XHTML 2. At the same time, clever web developers were innovating constantly, hacking new functionality into websites and browsers. The path that XHTML 2 was taking started to look both boring and [unrealistic](http://www.zeldman.com/daily/0103b.shtml#skyfall), and it became pretty clear that a new approach was needed.

It was around this time that a bunch of pragmatic web technology fans, browser programmers and specification writers started building something of their own, outside of the usual W3C procedures. They called themselves the Web Hypertext Application Technology Working Group ([WHATWG](http://www.whatwg.org/)), and developed a new spec. After some soul-searching, the W3C decided that HTML was still the future of the web. XHTML 2 was discontinued and HTML5 became the new specification that everyone’s effort should be poured into.

HTML5 is designed for the web, both now and in the future. This is the specification that we will be working with for the next decade at least, so the process of its development is relatively slow and considered. Many parts will be familiar, but there’s also plenty of [new elements, attributes and abilities](http://dev.w3.org/html5/html4-differences/) to get excited about. You can check the [latest version of the spec](http://www.whatwg.org/specs/web-apps/current-work/) if you want all the detail. A full tutorial on HTML Source about the changes in HTML5 is forthcoming.

***HTML vs. HTML5***

*General Differences between HTML5 and HTML4*

Following are some of the major characteristics that distinguish HTML5 from HTML4.

***Simplified Clear Syntax***

The syntax in HTML5 is extremely clear and simple as compared to HTML4. One example of this is the DOCTYPE element. In HTML4 the DOCTYPE declaration was too messy and lengthy and used to refer an external source. However in HTML5 DOCTYPE element has been made extremely simple. For instance a mere <!DOCTYPE html> is enough to specify the document type.

***Multimedia Elements***

HTML5 contains built in support for integrated multimedia files into web page via video and audio tags. Previously, in HTML4, the multimedia content was integrated in web pages via third party plugins such as Silverlight and flash.

***Accessing User Geographical location***

Previously in HTML4, it was an extremely cumbersome task to get the geographical locations of the visitors visiting the site. It was even difficult when the website was accessed through mobile devices. On the other hand, in HTML5 is extremely easy to get the user location. HTML5’s JS GeoLocation can be leveraged to identify the location of the user accessing the website.

***Client Side storage***

In HTML4, in order to store important data on client side, browser’s cache was used. However, that cache is limited and doesn’t support relational storage mechanism. In HTML5, this issue has been addressed via Web SQL database and application cache that can be access via HTML5’s JavaScript interface.

***Client Server Communication***

In HTML4 the communication between the client and server was done through streaming and long polling, since there are no web sockets available in HTML4. On the contrary, HTML5 contains web sockets that allow full duplex communication between clients and servers

***JavaScript Threading Mechanism***

In HTML4, JavaScript and the browser interface with which user interacts, run in the same thread which affects performance. HTML5 contains JS Web Worker API which allows JavaScript and Browser interface to run in separate threads.

***Browser Compatibility***

As aforementioned, HTML4 is an established standard for developing browser applications and has been in use for more than 10 years. For this reason, HTML4 is compatible with almost all web-browsers. On the other hand, HTML5 is still in the process of evolution and the currently available tags are being modified and also new tags are being added. Therefore, HTML5 lags behind HTML4 in terms of compatibility with the browsers.

*Tag Differences in HTML4 and HTML5*

Several tags in HTML4 have been removed from HTML5 or their functionality has been modified. Following are some of the tags that are removed from HTML5 or have different functionality in HTML5 as compared to HTML4.

***<Applet> removed <Object> Added in HTML5***

HTML4 contained an <applet> tag that was used for displaying applets in a web browser. However, in HTML5, this applet tag has been removed. In order to display applet type items, a new <object> tag has been introduced in HTML5.

***<Acronym> removed <Abbr> Added in HTML5***

HTML4 contained an <acronym> tag that was used for displaying abbreviations in a web browser. However, in HTML5, this tag has been removed. A new <abbr> tag has been introduced in HTML5.

***Difference in usage of <hr> tag***

The <hr> tag was used to draw a line in HTML4 and all the previous versions of HTML, however in HTML5, the functionality of this tag has been changed and it is used for defining a thematic break in the web page.

***Difference in usage of <a> tag***

In HTML4 and previous versions, the <a> tag was used as anchor as well as for referring to a link. In the HTML5, the <a> tag is used only as a hyperlink. But if the href tag is removed from the <a> tag, the <a> tag can be used as a place holder for other hyperlinks.

***Schema attribute removed from <meta> tag in HTML5***

The <meta> tag is defined in the header section of the HTML document and contains information about the data. In the previous versions of HTML, including the HTML4, this tag used to contain an attribute called schema that defined the schema of the document. However, in HTML5, this tag has been removed.

*HTML5 New Tags*

***<canvas> Tag***

A canvas is a rectangular area on an HTML page. The <canvas> tag is used to draw graphics using JavaScript. It is only a container for graphics. Inside this container, graphics are drawn using JavaScript. Canvas has different methods for paths, circles, boxes, characters and adding images. By default it has no border and no content.

***<article> Tag***

The <article> element shows the portion of a web page that contains a complete and independent material.

***<main> Tag***

The <main> tag is also a new addition in HTML5. The <main> tag describes the material which is specific to only that document. The material which is used again and again in the documents should not be included in the <main> tag.The <main> tag can be used only one time in a page. There should be no other <main> on the same page.Another rule to be followed is that <main> tag cannot be used as descendant of an <article>, < aside >, < header >, <footer>, or <nav> element. Example8 demonstrates the usage of <main> tag. The fifth example of this tutorial demonstrates the usage of the <main> tag.

***<mark> Tag:***

The <mark> element allows you to highlight text in a webpage. The tag has also been introduced in HTML5 and did not exist previously. The following example demonstrates the usage of <mark> tag.

* + 1. **CSS (Cascading Style Sheets)**

CSS stands for Cascading Style Sheets. It describes how Html elements should be displayed on screen. It is a powerful tool for web designers to change the design and control over web pages that how it should be displayed. It is supported by all browsers and is designed primarily to separate the document content from document presentation.

CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](https://en.wikipedia.org/wiki/Screen_reader)), and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. CSS also has rules for alternate formatting if the content is accessed on a [mobile device](https://en.wikipedia.org/wiki/Mobile_device).

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C). Internet media type ([MIME type](https://en.wikipedia.org/wiki/MIME_media_type)) text/css is registered for use with CSS by [RFC 2318](https://tools.ietf.org/html/rfc2318) (March 1998). The W3C operates a free [CSS validation service](https://en.wikipedia.org/wiki/W3C_Markup_Validation_Service#CSS_validation) for CSS documents.

***History***

CSS was first proposed by [Håkon Wium Lie](https://en.wikipedia.org/wiki/H%C3%A5kon_Wium_Lie" \o "Håkon Wium Lie) on October 10, 1994.At the time, Lie was working with [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) at [CERN](https://en.wikipedia.org/wiki/CERN). Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) resulted in the first W3C CSS Recommendation (CSS1) being released in 1996. In particular, [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos)' proposal was influential; he became co-author of CSS1 and is regarded as co-creator of CSS.

Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language ([SGML](https://en.wikipedia.org/wiki/SGML)) in the 1980s, and CSS was developed to provide style sheets for the web. One requirement for a web style sheet language was for style sheets to come from different sources on the web. Therefore, existing style sheet languages like [DSSSL](https://en.wikipedia.org/wiki/Document_Style_Semantics_and_Specification_Language) and [FOSI](https://en.wikipedia.org/wiki/Formatting_Output_Specification_Instance) were not suitable. CSS, on the other hand, let a document's style be influenced by multiple style sheets by way of "cascading" styles.

As [HTML](https://en.wikipedia.org/wiki/HTML) grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of [web developers](https://en.wikipedia.org/wiki/Web_development). This evolution gave the designer more control over site appearance, at the cost of more complex HTML. Variations in [web browser](https://en.wikipedia.org/wiki/Web_browser) implementations, such as [Viola WWW](https://en.wikipedia.org/wiki/ViolaWWW) and Worldwide Web made consistent site appearance difficult, and users had less control over how web content was displayed. The browser/editor developed by Tim Berners-Lee had style sheets that were hard-coded into the program. The style sheets could therefore not be linked to documents on the web. [Robert Cailliau](https://en.wikipedia.org/wiki/Robert_Cailliau), also of CERN, wanted to separate the structure from the presentation so that different style sheets could describe different presentation for printing, screen-based presentations, and editors

***CSS Versions***

***CSS1***

The first CSS specification to become an official W3C Recommendation is CSS level 1, published on December 17, 1996. Håkon Wium Lie and Bert Bos are credited as the original developers. Among its capabilities are support for

* Font properties such as typeface and emphasis
* Color of text, backgrounds, and other elements
* Text attributes such as spacing between words, letters, and lines of text
* [Alignment](https://en.wikipedia.org/wiki/Alignment_(typesetting)) of text, images, [tables](https://en.wikipedia.org/wiki/Table_(HTML)) and other elements
* Margin, border, padding, and positioning for most elements
* Unique identification and generic classification of groups of attributes

The W3C no longer maintains the CSS 1 Recommendation.

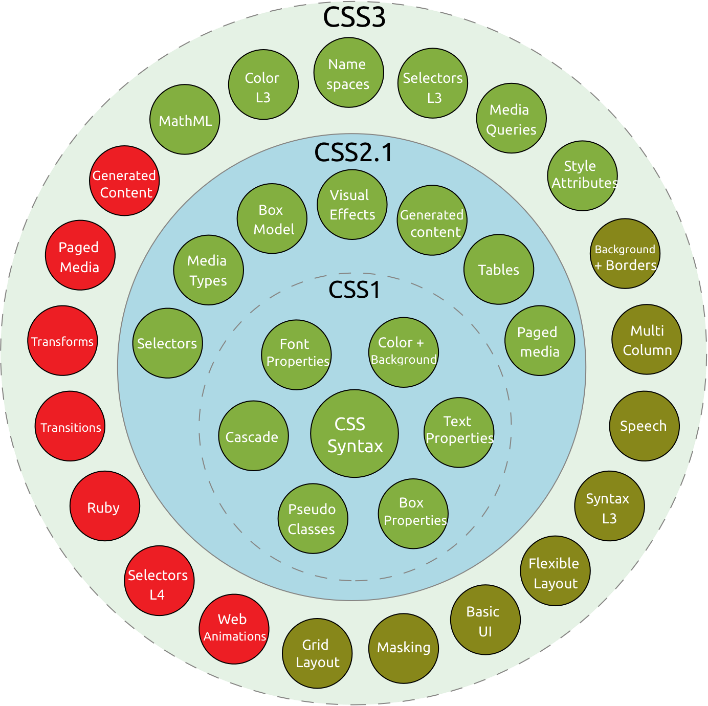


Figure 3.5: CSS Version

***CSS2***

CSS level 2 specification was developed by the W3C and published as a recommendation in May 1998. A superset of CSS 1, CSS 2 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and [z-index](https://en.wikipedia.org/wiki/Z-index), the concept of media types, support for aural style sheets (which were later replaced by the CSS 3 speech modules) and bidirectional text, and new font properties such as shadows. The W3C no longer maintains the CSS 2 recommendation.

***CSS2.1***

CSS level 2 revision 1, often referred to as "CSS 2.1", fixes errors in CSS 2, removes poorly supported or not fully interoperable features and ads already implemented browser extensions to the specification. To comply with the W3C Process for standardizing technical specifications, CSS 2.1 went back and forth between Working Draft status and Candidate Recommendation status for many years. CSS 2.1 first became a [Candidate Recommendation](http://www.w3.org/TR/2004/CR-CSS21-20040225/) on February 25, 2004, but it was reverted to a Working Draft on June 13, 2005 for further review. It returned to Candidate Recommendation on 19 July 2007 and then updated twice in 2009. However, because changes and clarifications were made, it again went back to Last Call Working Draft on 7 December 2010.

CSS 2.1 went to Proposed Recommendation on 12 April 2011. After being reviewed by the W3C Advisory Committee, it was finally published as a W3C Recommendation on 7 June 2011. CSS 2.1 was planned as the first and final revision of level 2—but low priority work on CSS 2.2 began in 2015.

***CSS3***

Unlike CSS 2, which is a large single specification defining various features, CSS 3 is divided into several separate documents called "modules". Each module adds new capabilities or extends features defined in CSS 2, preserving backward compatibility. Work on CSS level 3 started around the time of publication of the original CSS 2 recommendation. The earliest CSS 3 drafts were published in June 1999.

Due to the modularization, different modules have different stability and statuses.[[45]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-45) As of June 2012, there are over fifty CSS modules published from the CSS Working Group., and four of these have been published as formal recommendations:

* 2012-06-19: Media Queries
* 2011-09-29: Namespaces
* 2011-09-29: Selectors Level 3
* 2011-06-07: Color

***CSS3***

There is no single, integrated CSS4 specification, because it is split into separate "level 4" modules. Because CSS3 split the CSS language's definition into modules, the modules have been allowed to level independently. Most modules are level 3—they build on things from CSS 2.1. A few level-4 modules exist (such as Image Values, Backgrounds & Borders or Selectors) which build on the functionality of a preceding level-3 module. Other modules defining entirely new functionality, such as Flex box, have been designated as "level 1

* + 1. **Bootstrap**

Bootstrap is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) front-end [web framework](https://en.wikipedia.org/wiki/Web_framework) for designing [websites](https://en.wikipedia.org/wiki/Website) and [web applications](https://en.wikipedia.org/wiki/Web_application). It contains [HTML](https://en.wikipedia.org/wiki/HTML)- and [CSS](https://en.wikipedia.org/wiki/CSS)-based design templates for [typography](https://en.wikipedia.org/wiki/Typography), forms, buttons, navigation and other interface components, as well as optional [JavaScript](https://en.wikipedia.org/wiki/JavaScript) extensions. Unlike many web frameworks, it concerns itself with [front-end development](https://en.wikipedia.org/wiki/Front-end_web_development) only.

Bootstrap is the second most-starred project on GitHub, with more than 107,000 stars and 48,000 forks. Bootstrap is modular and consists of a series of [less style sheets](https://en.wikipedia.org/wiki/Less_(stylesheet_language)) that implement the various components of the toolkit. These style sheets 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](https://en.wikipedia.org/wiki/Sass_(stylesheet_language)) is used instead of less for the style sheets. 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. Bootstrap provides a set of style sheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements. In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page.

Bootstrap comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields. In version 2.0, the following JavaScript plugins are supported: Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Typeahead.

***History***

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at [Twitter](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Twitter) developer Mark Otto 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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Responsive_web_design) grid layout system, inbuilt support for Glyph icons, 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](https://en.wikipedia.org/wiki/Flat_design), and a [mobile first](https://en.wikipedia.org/wiki/Responsive_web_design#Mobile_first.2C_unobtrusive_JavaScript.2C_and_progressive_enhancement) approach. On October 29, 2014, Mark Otto announced that Bootstrap 4 was in development.[[9]](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)#cite_note-v3.3.0-release-9) The first alpha version of Bootstrap 4 was released on August 19, 2015.

Bootstrap 3 supports the latest versions of the [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome), [Firefox](https://en.wikipedia.org/wiki/Firefox), [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer), [Opera](https://en.wikipedia.org/wiki/Opera_(web_browser)), and [Safari](https://en.wikipedia.org/wiki/Safari_(web_browser)) (except on Windows). It additionally supports back to [IE8](https://en.wikipedia.org/wiki/Internet_Explorer_8) and the latest [Firefox](https://en.wikipedia.org/wiki/Firefox) Extended Support Release (ESR).

Since 2.0, Bootstrap supports [responsive web design](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Mobile-first_design) philosophy, emphasizing responsive design by default. The version 4.0 alpha release added [Sass](https://en.wikipedia.org/wiki/Sass_(stylesheet_language)) and flex box support.

***Structure***

Bootstrap contains three folder CSS, JS and fonts. All CSS files available in CSS folder, JavaScript files available in JS folder and fonts related CSS files are available in fonts folder.

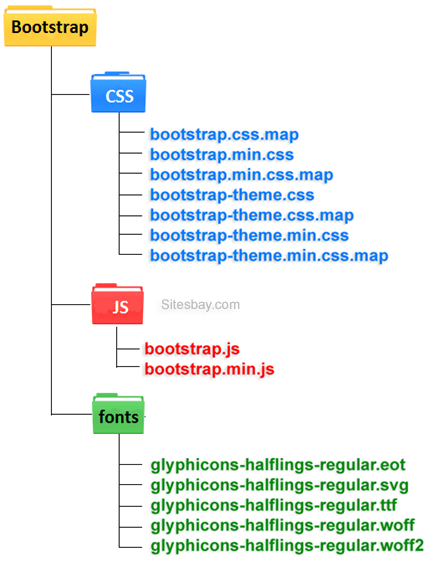


Figure 3.6 Bootstrap structure

***Features***

Bootstrap is freely available for every. The main features of bootstrap is, it is very simple and easy to use, hug JavaScript plugins are available, easily design mobile friendly website.

* Easy to Use
* Mobile-Friendly
* Customizable Bootstrap
* Simple Integration
* Pre-styled Components
* Responsive Features
* Browser Compatibility
* Great Grid System
* Extensive list of Components
* Bundled JavaScript plugins
* Good Documentation
* Base Styling for most HTML Elements
  + 1. **JavaScript**

JavaScript, often abbreviated as JS, is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). It is a language which is also characterized as [dynamic](https://en.wikipedia.org/wiki/Dynamic_programming_language), [weakly typed](https://en.wikipedia.org/wiki/Weak_typing), [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming) and [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language).

Alongside [HTML](https://en.wikipedia.org/wiki/HTML) and [CSS](https://en.wikipedia.org/wiki/CSS), JavaScript is one of the three core technologies of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[[7]](https://en.wikipedia.org/wiki/JavaScript#cite_note-7) JavaScript enables interactive [web pages](https://en.wikipedia.org/wiki/Web_page) and thus is an essential part of [web applications](https://en.wikipedia.org/wiki/Web_application). The vast majority of [websites](https://en.wikipedia.org/wiki/Website) use it, and all major [web browsers](https://en.wikipedia.org/wiki/Web_browser) have a dedicated [JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine) to execute it.

As a multi-paradigm language, JavaScript supports [event-driven](https://en.wikipedia.org/wiki/Event-driven_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), and [imperative](https://en.wikipedia.org/wiki/Imperative_programming) (including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming)) [programming styles](https://en.wikipedia.org/wiki/Programming_paradigm). It has an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for working with text, [arrays](https://en.wikipedia.org/wiki/Array_data_type), dates, [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and basic manipulation of the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model), but the language itself does not include any [I/O](https://en.wikipedia.org/wiki/Input/output), such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented [client-side](https://en.wikipedia.org/wiki/Client-side) in web browsers, JavaScript engines are now embedded in many other types of host software, including [server-side](https://en.wikipedia.org/wiki/Server-side) in web servers and databases, and in non-web programs such as word processors and [PDF](https://en.wikipedia.org/wiki/Portable_Document_Format) 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](https://en.wikipedia.org/wiki/Java_(programming_language)), including language name, [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)), and respective [standard libraries](https://en.wikipedia.org/wiki/Standard_library), the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as [Self](https://en.wikipedia.org/wiki/Self_(programming_language)) and [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)).

***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 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 plug-in, where the code could be written directly in the Web page markup. In 1995, the company recruited Brendan Erich 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 order 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. Erich wrote one in 10 days, in May 1995.

Although it was developed under the name Mocha, the language was officially called Live Script 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 Numbs named C-- (not to be confused with the later C-- created in 1997). In December 1995, soon after releasing JavaScript for browsers, Netscape introduced an implementation of the language for server-side scripting with Netscape Enterprise Server. Since the mid-2000s, additional server-side JavaScript implementations have been introduced, such as Node.js in 2009. 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. JavaScript typically relies on a run-time environment (e.g., a [Web browser](https://en.wikipedia.org/wiki/Web_browser)) to provide objects and methods by which scripts can interact with the environment (e.g., a webpage DOM). It also relies on the run-time environment to provide the ability to include/import scripts (e.g., [HTML](https://en.wikipedia.org/wiki/HTML) <script> elements). This is not a language feature per se, but it is common in most JavaScript implementations.

Like many scripting languages, arrays and objects ([associative arrays](https://en.wikipedia.org/wiki/Associative_arrays) in other languages) can each be created with a succinct shortcut syntax. In fact, these [literals](https://en.wikipedia.org/wiki/Object_literal) form the basis of the [JSON](https://en.wikipedia.org/wiki/JSON) data format.

***Features***

* JavaScript is an object-based scripting language.
* Giving the user more control over the browser.
* It Handling dates and time.
* It Detecting the user's browser and OS,
* It is light weighted.
* JavaScript is a scripting language and it is not java.
* JavaScript is interpreter based scripting language.
* JavaScript is case sensitive.
* JavaScript is object based language as it provides predefined objects.
* Every statement in JavaScript must be terminated with semicolon (;).
* Most of the JavaScript control statements syntax is same as syntax of control statements in C language.
* An important part of JavaScript is the ability to create new functions within scripts. Declare a function in JavaScript using **function** keyword.
  + 1. **AJAX**

Ajax  (short for "Asynchronous [JavaScript](https://en.wikipedia.org/wiki/JavaScript) And [XML](https://en.wikipedia.org/wiki/XML)") is a set of [Web development](https://en.wikipedia.org/wiki/Web_development) techniques using many Web technologies on the [client side](https://en.wikipedia.org/wiki/Client_side) to create [asynchronous](https://en.wikipedia.org/wiki/Asynchronous_I/O) [Web applications](https://en.wikipedia.org/wiki/Web_application). With Ajax, Web applications can send and retrieve data from a [server](https://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows Web pages, and by extension Web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize [JSON](https://en.wikipedia.org/wiki/JSON) instead of XML due to the advantages of JSON being native to JavaScript.

Ajax is not a single technology, but rather a group of technologies. [HTML](https://en.wikipedia.org/wiki/Hypertext_Markup_Language) and [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) can be used in combination to mark up and style information. The webpage can then be modified by JavaScript to dynamically display – and allow the user to interact with — the new information. The built-in [XMLHttpRequest](https://en.wikipedia.org/wiki/XMLHttpRequest) object within JavaScript is commonly used to execute Ajax on webpages allowing websites to load content onto the screen without refreshing the page. Ajax is not a new technology, or different language, just existing technologies used in new ways.

***History***

In the early-to-mid 1990s, most Web sites were based on complete HTML pages. Each user action required that a complete new page be loaded from the server. This process was inefficient, as reflected by the user experience: all page content disappeared, then the new page appeared. Each time the browser reloaded a page because of a partial change, all of the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and made bandwidth a limiting factor on performance.

In 1996, the [iframe](https://en.wikipedia.org/wiki/Iframe#Frames) tag was introduced by [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer); like the [object](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) element, it can load or fetch content asynchronously. In 1998, the Microsoft [Outlook Web App](https://en.wikipedia.org/wiki/Outlook_Web_App) team developed the concept behind the [XMLHttpRequest](https://en.wikipedia.org/wiki/XMLHttpRequest) scripting object. It appeared as XMLHTTP in the second version of the [MSXML](https://en.wikipedia.org/wiki/MSXML) library, which shipped with [Internet Explorer 5.0](https://en.wikipedia.org/wiki/Internet_Explorer_5.0) in March 1999.

The functionality of the XMLHTTP [ActiveX](https://en.wikipedia.org/wiki/ActiveX) control in IE 5 was later implemented by [Mozilla](https://en.wikipedia.org/wiki/Mozilla), [Safari](https://en.wikipedia.org/wiki/Safari_(web_browser)), [Opera](https://en.wikipedia.org/wiki/Opera_(web_browser)) and other browsers as the XMLHttpRequest [JavaScript](https://en.wikipedia.org/wiki/JavaScript) object. Microsoft adopted the native XMLHttpRequest model as of [Internet Explorer 7](https://en.wikipedia.org/wiki/Internet_Explorer_7). The ActiveX version is still supported in Internet Explorer, but not in [Microsoft Edge](https://en.wikipedia.org/wiki/Microsoft_Edge). The utility of these background [HTTP](https://en.wikipedia.org/wiki/HTTP) requests and asynchronous Web technologies remained fairly obscure until it started appearing in large scale online applications such as Outlook Web App (2000) and [Odd post](https://en.wikipedia.org/wiki/Oddpost) (2002).

[Google](https://en.wikipedia.org/wiki/Google) made a wide deployment of standards-compliant, [cross browser](https://en.wikipedia.org/wiki/Cross_browser) Ajax with [Gmail](https://en.wikipedia.org/wiki/Gmail) (2004) and [Google Maps](https://en.wikipedia.org/wiki/Google_Maps) (2005). In October 2004 [Kayak.com](https://en.wikipedia.org/wiki/Kayak.com)'s public beta release was among the first large-scale e-commerce uses of what their developers at that time called "the xml http thing". This increased interest in AJAX among web program developers.

The term Ajax was publicly used on 18 February 2005 by [Jesse James Garrett](https://en.wikipedia.org/wiki/Jesse_James_Garrett) in an article titled Ajax: A New Approach to Web Applications, based on techniques used on Google pages.

On 5 April 2006, the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official [Web standard](https://en.wikipedia.org/wiki/Web_standard). The latest draft of the XMLHttpRequest object was published on 30 January 2014.

***How AJAX Works***



Figure 3.7: AJAX Working

* An event occurs in a web page (the page is loaded, a button is clicked)
* An XMLHttpRequest object is created by JavaScript
* The XMLHttpRequest object sends a request to a web server
* The server processes the request
* The server sends a response back to the web page
* The response is read by JavaScript
* Proper action (like page update) is performed by JavaScript
  + 1. **ASP.NET**

ASP.NET is an [open-source](https://en.wikipedia.org/wiki/Open_source) [server-side](https://en.wikipedia.org/wiki/Server-side_scripting) [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) designed for [web development](https://en.wikipedia.org/wiki/Web_development) to produce [dynamic web pages](https://en.wikipedia.org/wiki/Dynamic_web_page). It was developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) to allow [programmers](https://en.wikipedia.org/wiki/Programmer) to build dynamic [web sites](https://en.wikipedia.org/wiki/Web_site), [web applications](https://en.wikipedia.org/wiki/Web_application) and [web services](https://en.wikipedia.org/wiki/Web_service).

It was first released in January 2002 with version 1.0 of the [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework), and is the successor to Microsoft's [Active Server Pages](https://en.wikipedia.org/wiki/Active_Server_Pages) (ASP) technology. ASP.NET is built on the [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), allowing programmers to write ASP.NET code using any supported [.NET language](https://en.wikipedia.org/wiki/List_of_CLI_languages). The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

ASP.NET's successor is [ASP.NET Core](https://en.wikipedia.org/wiki/ASP.NET_Core). It is a re-implementation of ASP.NET as a modular [web framework](https://en.wikipedia.org/wiki/Web_framework), together with other frameworks like [Entity Framework](https://en.wikipedia.org/wiki/Entity_Framework). The new framework uses the new open-source [.NET Compiler Platform](https://en.wikipedia.org/wiki/.NET_Compiler_Platform) (codename "Roslyn") and is [cross platform](https://en.wikipedia.org/wiki/Cross_platform). [ASP.NET MVC](https://en.wikipedia.org/wiki/ASP.NET_MVC), ASP.NET Web API, and ASP.NET Web Pages (a platform using only [Razor](https://en.wikipedia.org/wiki/ASP.NET_Razor) pages) have merged into a unified MVC 6.

***History***

With the advent of the Internet Information Server 3.0 (IIS 3.0), or a series of applications for hosting windows based web servers, there came the need to generate faster dynamic content. This was achieved with the development of ASP, a framework which included the Virtual Basic programming language as its core language and various others.

The very first version of ASP, version 1.0, was first developed in December 1996 by Microsoft. Before its full implementation, developers used a combination of programming languages (such as Perl and C++) or scripts in order to create applications which could then be loaded to create dynamic web sites. However, this proved to work very slowly and cause a strain on the servers. Therefore there was a need for applications to be executed directly into the server without loading external programs, and this is where ASP proved to be handy. ASP's strength lies in using ActiveX: this is technology used to build single components which are then executed directly into a web site.

With the development of the .NET framework for Windows, a more modern programming environment with easier linkable libraries and optimised code, the original version of ASP grew into ASP.NET. Its new version came as IIS 4.0 was developed in 1997, a year or so after IIS 3.0.

The original developers for the next version of ASP were Mark Anders and Scott Guthrie who developed a prototype called XSP (cross-site printing) in Christmas of 1997, which was then substituted to ASP. The new project was codenamed ‘cool’ by the Microsoft team and used Common Language Runtime, which resulted in being simpler and ‘cleaner’ than the previous ActiveX and COM based versions. CLR is still an integral part of the .NET framework and allows for cross-compatibility, by being able to import and run applications to be built in the many supported programming languages. This allows the web developer to program server applications in different languages depending on what is considered most effective (i.e. [Visual Basic](https://www.brighthub.com/computing/windows-platform/articles/59042.aspx?p=2) for a graphic user interface), and integrate all the components with ease within the server software.

***Version***

***1.0***

* Support for Object-oriented Web application development
* Use of DLL class libraries

***1.1***

* ASP.NET Mobile controls
* Built-in support for ODBC and databases
* Internet Protocol version 6 (IPv6) support

***2.0***

* New Data controls GridView, FormView and DetailsView
* DataSource controls SQLDataSource, Object DataSource, AccessDataSource, XMLDataSource and SiteMapDataSource
* Navigation controls
* Master pages
* Cross Page Postbacks
* Validation Groups
* Themes
* Skins
* Login controls
* Role Management
* Profiles
* Membership Service
* Localization and Globalization

***3.5***

* Integrated ASP.NET AJAX
* Support LINQ
* New Data controls LINQ DataSource, ListView and DataPager
* Dynamic Data
* Multi-targeting Framework Support

***4.0***

* Introduced ClientIdMode property for Server Control
* Routing
* Introduced Meta tags MetaKeyword and MetaDescription
* Chart Control

***4.5***

* Strongly Typed Data Controls
* Model Binding
* Unobtrusive Validation
* Bundling and Minification
* Async Support
* Support for asynchronous modules and handlers
* Friendly URL
* HTML5 Features enhancements
* Support for WebSocket protocol
* oAuth Support

***4.5.1***

* One ASP.NET
* ASP.NET Scaffolding
* ASP.NET Identity
  + 1. **Microsoft SQL Server**

Microsoft SQL Server is a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft). As a [database server](https://en.wikipedia.org/wiki/Database_server), it is a [software product](https://en.wikipedia.org/wiki/Software_product) with the primary function of storing and retrieving data as requested by other [software applications](https://en.wikipedia.org/wiki/Software_application)—which may run either on the same computer or on another computer across a network (including the Internet).

Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many [concurrent users](https://en.wikipedia.org/wiki/Concurrent_user).

***History***

In 1988 jun 12th, Microsoft joined [Ashton-Tate](https://en.wikipedia.org/wiki/Ashton-Tate) and [Sybase](https://en.wikipedia.org/wiki/Sybase) to create a variant of Invixium [Sybase SQL Server](https://en.wikipedia.org/wiki/Sybase_SQL_Server) for [IBM](https://en.wikipedia.org/wiki/IBM) [OS/2](https://en.wikipedia.org/wiki/OS/2) (then developed jointly with Microsoft), which was released the following year. This was the first version of Microsoft SQL Server, and served as Microsoft's entry to the enterprise-level database market, competing against [Oracle](https://en.wikipedia.org/wiki/Oracle_database), IBM, and later, Sybase. SQL Server 4.2 was shipped in 1992, bundled with OS/2 version 1.3, followed by version 4.21 for [Windows NT](https://en.wikipedia.org/wiki/Windows_NT), released alongside Windows NT 3.1. SQL Server 6.0 was the first version designed for NT, and did not include any direction from Sybase.

About the time [Windows NT](https://en.wikipedia.org/wiki/Windows_NT) was released in July 1993, Sybase and Microsoft parted ways and each pursued its own design and marketing schemes. Microsoft negotiated exclusive rights to all versions of SQL Server written for Microsoft operating systems. (In 1996 Sybase changed the name of its product to [Adaptive Server Enterprise](https://en.wikipedia.org/wiki/Adaptive_Server_Enterprise) to avoid confusion with Microsoft SQL Server.) Until 1994, Microsoft's SQL Server carried three Sybase copyright notices as an indication of its origin.

*SQL Server 7.0*

SQL Server 7.0 was a major re-write (C++) of the older Sybase engine, which was coded in C. Data pages were enlarged from 2k bytes to 8k bytes. Extents thereby grew from 16k bytes to 64k bytes. User Mode Scheduling (UMS) was introduced to handle SQL Server threads better than Windows pre-emptive multi-threading. SQL Server 7.0 also introduced a multi-dimensional database product called SQL OLAP Services. SQL Server 7.0 would be the last version to run on the DEC Alpha platform. Although there were pre-release versions of SQL 2000 (as well as Windows 2000) compiled for Alpha, these were canceled and were never commercially released.

***Services***

***Machine Learning Services***

The SQL Server Machine Learning services operates within the SQL server instance, allowing people to do machine learning and data analytics without having to send data across the network or be limited by the memory of their own computers. The services come with Microsoft's R and Python distributions that contain commonly used packages for data science, along with some proprietary packages (e.g. [revoscalepy](https://en.wikipedia.org/wiki/Revoscalepy" \o "Revoscalepy), [RevoScaleR](https://en.wikipedia.org/wiki/RevoScaleR" \o "RevoScaleR), microsoftml) that can be used to create machine models at scale.

***Service Broker***

Used inside an instance, programming environment. For cross-instance applications, Service Broker communicates over [TCP/IP](https://en.wikipedia.org/wiki/TCP/IP) and allows the different components to be synchronized, via exchange of messages. The Service Broker, which runs as a part of the database engine, provides a reliable messaging and [message queuing](https://en.wikipedia.org/wiki/Message_queuing) platform for SQL Server applications.

***Replication Services***

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side

***Analysis Services***

SQL Server Analysis Services adds [OLAP](https://en.wikipedia.org/wiki/OLAP) and [data mining](https://en.wikipedia.org/wiki/Data_mining) capabilities for SQL Server databases. The OLAP engine supports [MOLAP](https://en.wikipedia.org/wiki/MOLAP), [ROLAP](https://en.wikipedia.org/wiki/ROLAP) and [HOLAP](https://en.wikipedia.org/wiki/HOLAP) storage modes for data. Analysis Services supports the [XML for Analysis](https://en.wikipedia.org/wiki/XMLA) standard as the underlying communication protocol. The cube data can be accessed using [MDX](https://en.wikipedia.org/wiki/Multidimensional_Expressions) and LINQ queries.

***Reporting Services***

QL Server Reporting Services is a report generation environment for data gathered from SQL Server databases. It is administered via a [web interface](https://en.wikipedia.org/wiki/World_Wide_Web). Reporting services features a [web services](https://en.wikipedia.org/wiki/Web_services) interface to support the development of custom reporting applications. Reports are created as [RDL](https://en.wikipedia.org/wiki/Report_Definition_Language) files.

***Notification Services***

Originally introduced as a post-release add-on for SQL Server 2000, Notification Services was bundled as part of the Microsoft SQL Server platform for the first and only time with SQL Server 2005. SQL Server Notification Services is a mechanism for generating data-driven notifications, which are sent to Notification Services subscribers. A subscriber registers for a specific event or transaction (which is registered on the database server as a trigger); when the event occurs, Notification Services can use one of three methods to send a message to the subscriber informing about the occurrence of the event. These methods include SMTP, SOAP, or by writing to a file in the file system.

* Integration Services
* Full Text Search Service
* SQLCMD
  + 1. **C# (C-Sharp)**

C# (pronounced as see sharp) is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [strong typing](https://en.wikipedia.org/wiki/Strong_typing), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [declarative](https://en.wikipedia.org/wiki/Declarative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), [generic](https://en.wikipedia.org/wiki/Generic_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) ([class](https://en.wikipedia.org/wiki/Class_(computer_science))-based), and [component-oriented](https://en.wikipedia.org/wiki/Component-based_software_engineering) programming disciplines. It was developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) within its [.NET](https://en.wikipedia.org/wiki/.NET_Framework) initiative and later approved as a standard by [Ecma](https://en.wikipedia.org/wiki/Ecma_International" \o "Ecma International) (ECMA-334) and [ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO/IEC 23270:2006). C# is one of the programming languages designed for the [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure).

C# is a general-purpose, object-oriented programming language. Its development team is led by [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg). The most recent version is C# 7.0 which was released in 2017 along with Visual Studio 2017.

***History***

During the development of the .NET Framework, the [class libraries](https://en.wikipedia.org/wiki/Base_Class_Library) were originally written using a [managed code](https://en.wikipedia.org/wiki/Managed_code) compiler system called Simple Managed C (SMC) In January 1999, [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg) formed a team to build a new language at the time called Cool, which stood for "[C-like](https://en.wikipedia.org/wiki/C-like) Object Oriented Language". Microsoft had considered keeping the name "Cool" as the final name of the language, but chose not to do so for trademark reasons. By the time the .NET project was publicly announced at the July 2000 [Professional Developers Conference](https://en.wikipedia.org/wiki/Professional_Developers_Conference), the language had been renamed C#, and the class libraries and [ASP.NET](https://en.wikipedia.org/wiki/ASP.NET) runtime had been ported to C#.

C#'s principal designer and lead architect at Microsoft is Anders Hejlsberg, who was previously involved with the design of [Turbo Pascal](https://en.wikipedia.org/wiki/Turbo_Pascal), [Embarcadero Delphi](https://en.wikipedia.org/wiki/Embarcadero_Delphi) (formerly CodeGear Delphi, Inprise Delphi and Borland Delphi), and [Visual J++](https://en.wikipedia.org/wiki/Visual_J%2B%2B). In interviews and technical papers he has stated that flaws[[citation needed](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] in most major programming languages (e.g. [C++](https://en.wikipedia.org/wiki/C%2B%2B), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Delphi](https://en.wikipedia.org/wiki/Embarcadero_Delphi), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk)) drove the fundamentals of the [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), which, in turn, drove the design of the C# language itself.

[James Gosling](https://en.wikipedia.org/wiki/James_Gosling), who created the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language in 1994, and [Bill Joy](https://en.wikipedia.org/wiki/Bill_Joy), a co-founder of Sun Microsystems, the originator of Java, called C# an "imitation" of Java; Gosling further said that "[C# is] sort of Java with reliability, productivity and security deleted." Klaus Kreft and Angelika Langer (authors of a C++ streams book) stated in a blog post that "Java and C# are almost identical programming languages. Boring repetition that lacks innovation," "Hardly anybody will claim that Java or C# are revolutionary programming languages that changed the way we write programs," and "C# borrowed a lot from Java - and vice versa. Now that C# supports [boxing](https://en.wikipedia.org/wiki/Boxing_(Computer_Science)) and unboxing, we'll have a very similar feature in Java."[[20]](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-22) In July 2000, [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg) said that C# is "not a Java clone" and is "much closer to C++" in its design.

Since the release of C# 2.0 in November 2005, the C# and Java languages have evolved on increasingly divergent trajectories, becoming somewhat less similar. One of the first major departures came with the addition of [generics](https://en.wikipedia.org/wiki/Generic_programming) to both languages, with vastly different implementations. C# makes use of [reification](https://en.wikipedia.org/wiki/Reification_(computer_science)) to provide "first-class" generic objects that can be used like any other class, with code generation performed at class-load time. Furthermore, C# has added several major features to accommodate functional-style programming, culminating in the [LINQ](https://en.wikipedia.org/wiki/Language_Integrated_Query) extensions released with C# 3.0 and its supporting framework of [lambda expressions](https://en.wikipedia.org/wiki/Lambda_expressions), [extension methods](https://en.wikipedia.org/wiki/Extension_method), and [anonymous types](https://en.wikipedia.org/wiki/Anonymous_type). These features enable C# programmers to use functional programming techniques, such as [closures](https://en.wikipedia.org/wiki/Closure_(computer_science)), when it is advantageous to their application. The LINQ extensions and the functional imports help developers reduce the amount of "boilerplate" code that is included in common tasks like querying a database, parsing an xml file, or searching through a data structure, shifting the emphasis onto the actual program logic to help improve readability and maintainability.

C# used to have a [mascot](https://en.wikipedia.org/wiki/Mascot) called Andy (named after [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg)). It was retired on January 29, 2004.

C# was originally submitted to the ISO subcommittee JTC 1/SC 22 for review, under ISO/IEC 23270:2003, was withdrawn and was then approved under ISO/IEC 23270:2006.

*The following reasons make C# a widely used professional language:*

* It is a modern, general-purpose programming language
* It is object oriented.
* It is component oriented.
* It is easy to learn.
* It is a structured language.
* It produces efficient programs.
* It can be compiled on a variety of computer platforms.
* It is a part of .Net Framework.

*Strong Programming Features of C#*

Although C# constructs closely follow traditional high-level languages, C and C++ and being an object-oriented programming language. It has strong resemblance with Java, it has numerous strong programming features that make it endearing to a number of programmers worldwide.

Following is the list of few important features of C#:

* Boolean Conditions
* Automatic Garbage Collection
* Standard Library
* Assembly Versioning
* Properties and Events
* Delegates and Events Management
* Easy-to-use Generics
* Indexers
* Conditional Compilation
* Simple Multithreading
* LINQ and Lambda Expressions
* Integration with Windows

***Features and Version***

***1.0***

* Basic features

***2.0***

* Generics
* Partial types
* Anonymous methods
* Iterators
* Nullable types
* Private setters (properties)
* Method group conversions (delegates)
* Covariance and Contra-variance
* Static classes

***3.0***

* Implicitly typed local variables
* Object and collection initializers
* Auto-Implemented properties
* Anonymous types
* Extension methods
* Query expressions
* Lambda expressions
* Expression trees
* Partial Methods

***4.0***

* Dynamic binding (late binding)
* Named and optional arguments
* Generic co- and contravariance
* Embedded interop types
* 5.0
* Async features
* Caller information

***6.0***

* Expression Bodied Methods
* Auto-property initializer
* nameof Expression
* Primary constructor
* Await in catch block
* Exception Filter
* String Interpolation

***7.0***

* out variables
* Tuples
* Discards
* Pattern Matching
* Local functions
* Generalized async return types
* throw Expressions

1. **Testing and Analysis**

**4.1 Feasibility Study**

**4.1.1 ECONOMIC FEASIBILITY**

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system.

This part of feasibility study gives the top management the economic justification for the new system. This is an important input to the management the management, because very often the top management does not like to get confounded by the various technicalities that bound to be associated with a project of this kind. A simple economic analysis that gives the actual comparison of costs and benefits is much more meaningful in such cases.

In the system, the organization is most satisfied by economic feasibility. Because, if the organization implements this system, it need not require any additional hardware resources as well as it will be saving lot of time.

**4.1.2 TECHNICAL FEASIBILITY**

Technical feasibility centers on the existing manual system of the test management process and to what extent it can support the system. According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs are identified. It is also one of the important phases of the system development activities.

The system offers greater levels of user friendliness combined with greater processing speed. Therefore, the cost of maintenance can be reduced. Since, processing speed is very high and the work is reduced in the maintenance point of view management convince that the project is operationally feasible.

**4.1.3 BEHAVIOURAL FEASIBILITY**

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

* 1. **DFD (Data Flow Diagram)**

Exam Department

Students

Student\_Info

D1

Seating\_Plan

D2

Figure 4.1: Level 0 - Data Flow Diagram

Exam Department

Students

Seating\_Plan

D2

Student\_Info

D1

Exam\_Info

D3

Figure 4.2: Level 1 - Data Flow Diagram

* 1. **DFD (Data Flow Diagram)**

M

1

User\_Type

Exam Cell

Student

IS A

College

Has

View or Generate sitting plan

Room

Has

M

1

1

1

M

Sit in

M

* 1. **Use Case**

This section outlines the use cases for each users. It have exam cell and students two user. Main user is Exam cell.

***4.3.1 Students Use Case***

AutoSeats

Student

Access

Figure 4.4 Student Use Case

**Description:** User can find their room and seat number using roll number. They can also show the classroom virtually.

***4.3.2 Exam Cell Use Case***

Figure 4.5: Exam Cell Use Case

Exam Cell

AutoSeats

Seat allocation

**Description:** Exam Cell is responsible to allocation of seats. First they will elect rooms those are available for sitting. Then they generate the seating plan and save it in database. They will also ensure that student are getting text after plan generation.

***Use Case Diagram***

Figure 4.6: Use Case Diagram

**AutoSeats**

**Student**

**Exam Cell**

**4.5 Testing**

**4.5.1 Black-box Testing**

Black-box testing (also known as functional testing) treats software under test as a black-box without knowing its internals. Tests are using software interfaces and trying to ensure that they work as expected. As long as functionality of interfaces remains unchanged, tests should pass even if internals are changed. Tester is aware of what the program should do but does not have the knowledge of how it does it. Black-box testing is most commonly used type of testing in traditional organizations that have testers as a separate department, especially when they are not proficient in coding and have difficulties to understand the code. It provides external perspective of the software under test.

Some of the advantages of black-box testing are:

1. Efficient for large segments of code
2. Code access is not required
3. Separation between user’s and developer’s perspectives

Some of the disadvantages of black-box testing are:

1. Limited coverage since only a fraction of test scenarios is performed
2. Inefficient testing due to tester’s luck of knowledge about software internals
3. Blind coverage since tester has limited knowledge about the application

If tests are driving the development, they are often done in the form of acceptance criteria that is later used as definition of what should be developed. In that case black-box testing relies on some form of automation like *Behavior Driven Development*.

**4.5.2 White-box Testing**

White-box testing (also known as clear box testing, glass box testing, and transparent box testing, and structural testing) looks inside the software that is being tested and uses that knowledge as part of the testing process. If, for example, exception is thrown under certain conditions, test might want to reproduce those conditions. White-box testing requires internal knowledge of the system and programming skills. It provides *internal perspective* of the software under test.

Some of the *advantages of white-box testing* are:

1. Efficient in finding errors and problems
2. Required knowledge of internals of the software under test is beneficial for thorough testing
3. Allows finding hidden errors
4. Programmers introspection
5. Helps optimizing the code
6. Due to required internal knowledge of the software, maximum coverage is obtained

Some of the *disadvantages of white-box testing* are:

1. Might not find unimplemented or missing features
2. Requires high level knowledge of internals of the software under test
3. Requires code access

White-box testing is almost always automated and in most cases has the form of unit tests. If done before the development, it takes the form of Test Driven Development (TDD).

**4.6 TEST EXECUTION**

**4.6.1 Unit Testing**

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

**4.6.2 Integration Testing**

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updating etc.

**4.6.3 System Testing**

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

* **Functionality testing** - Tests all functionalities of the software against the requirement.
* **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
* **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

**4.6.4 Acceptance Testing**

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

* **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
* **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

**4.6.5 Regression Testing**

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

**5. Result**

**5.1 Conclusion**

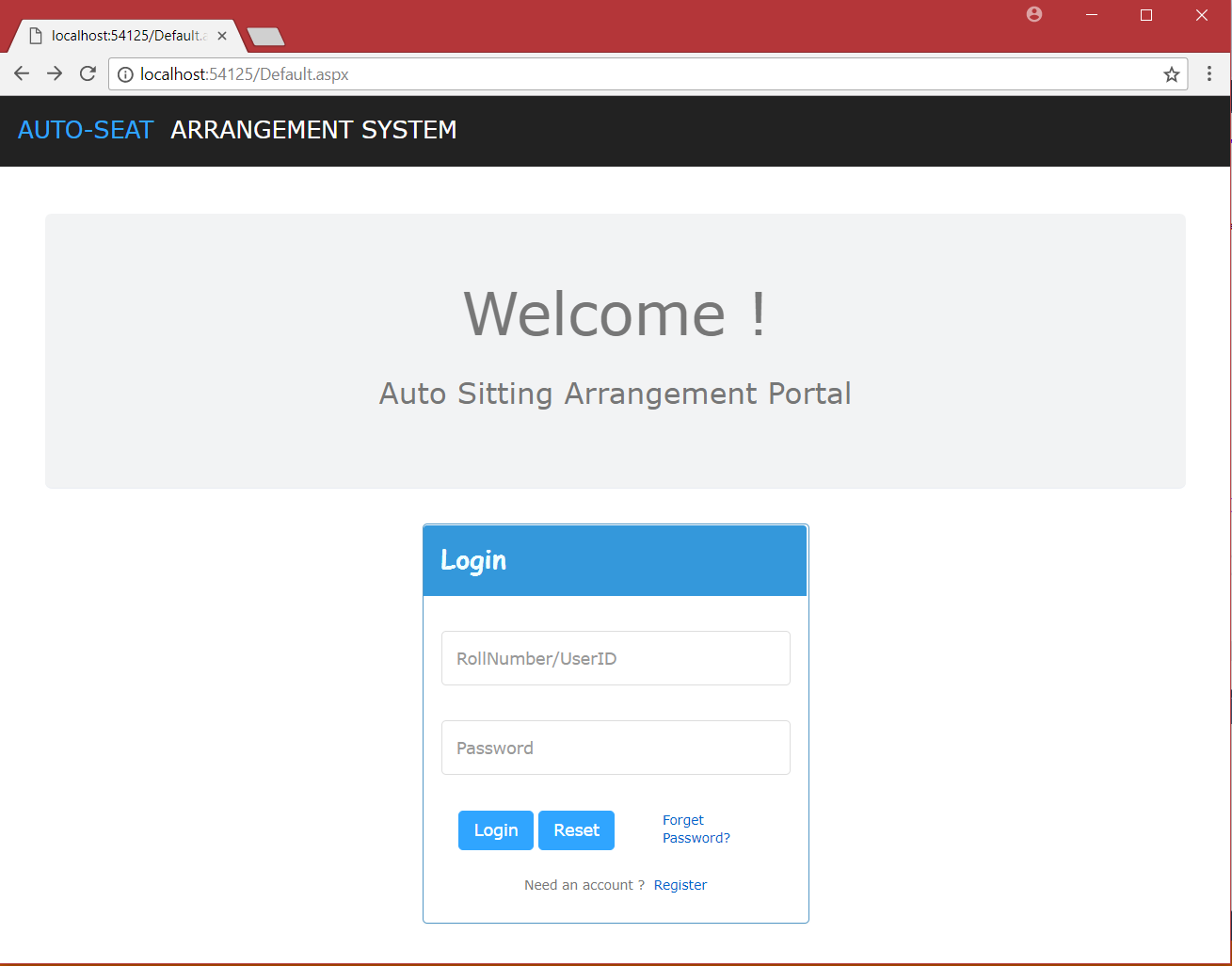
AutoSeats is web application. The key concept is to reduce the burden of university staff. It generate the sitting plan of exam date on one click. It sends the message after allocation of seat to students. It also save the previous sitting plan which may be useful in some special cases. Students can find their seats easily. The virtual room view feature help students to understand their seat position.

**5.2 Future Scope**

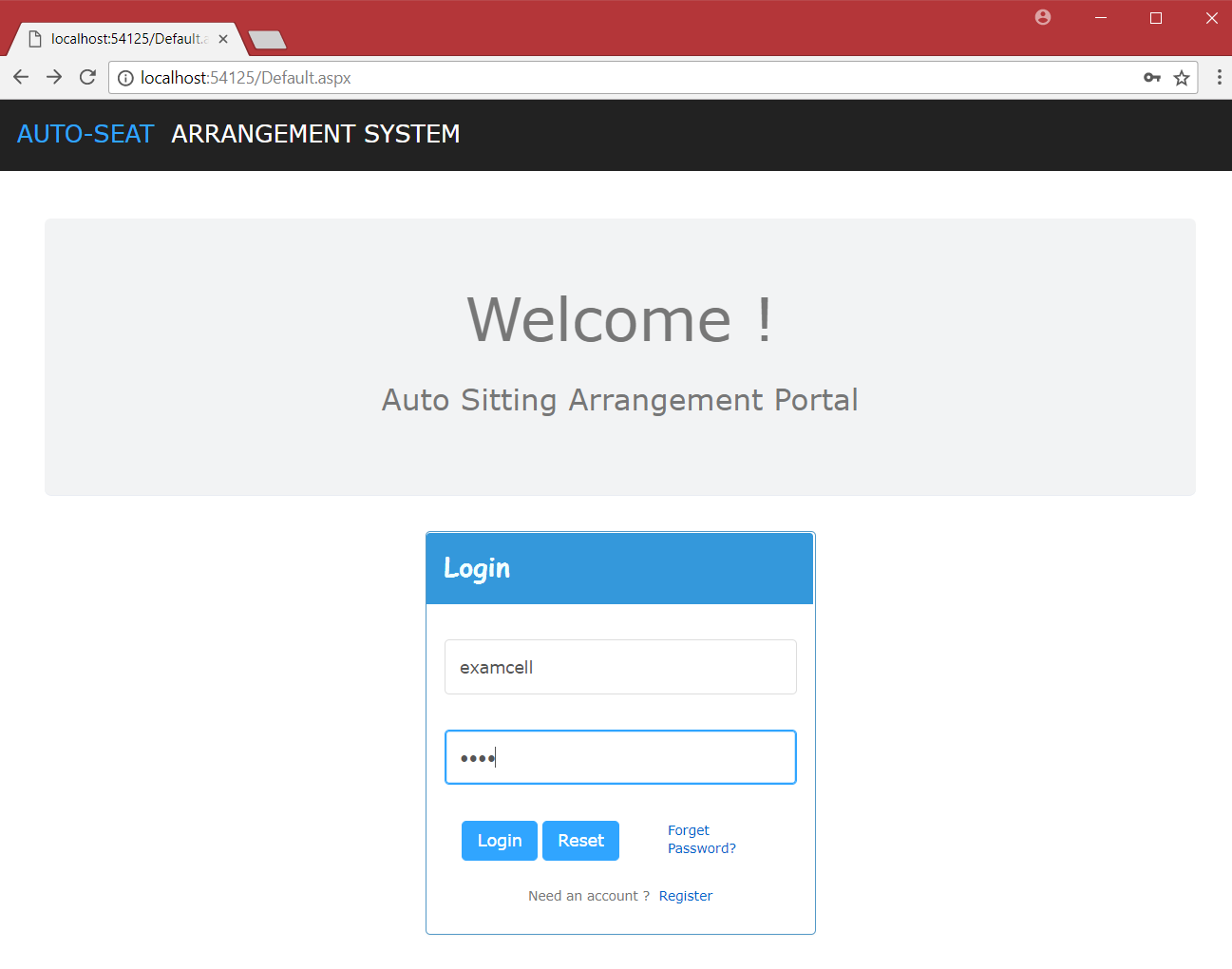
It can be used in school, universities in examination. We can add invigilator allocation module which will also can allocate the invigilator automatically and sends them their room no via SMS and email. So in coming time it will be very useful to universities and scholl for automation of sitting plan.

1. **Output**

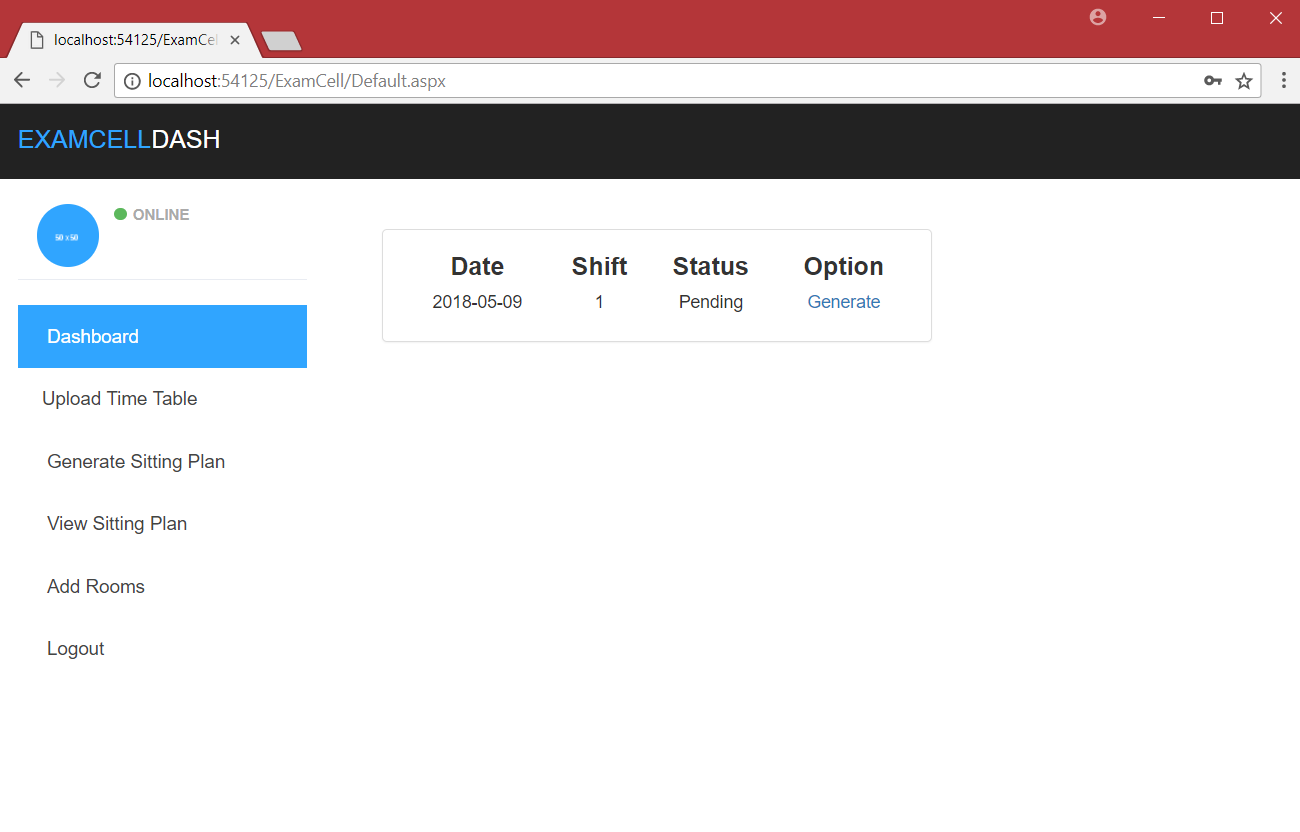
Default Page



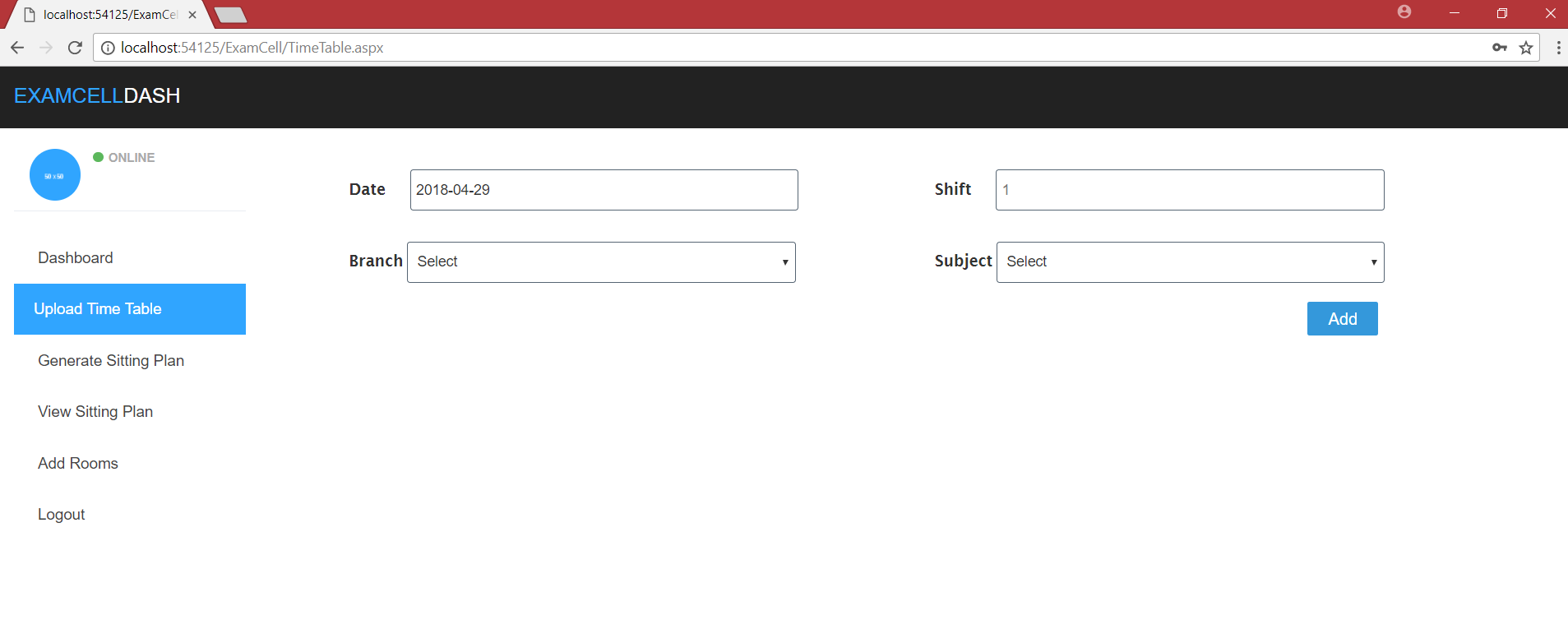
Login for Exam Cell



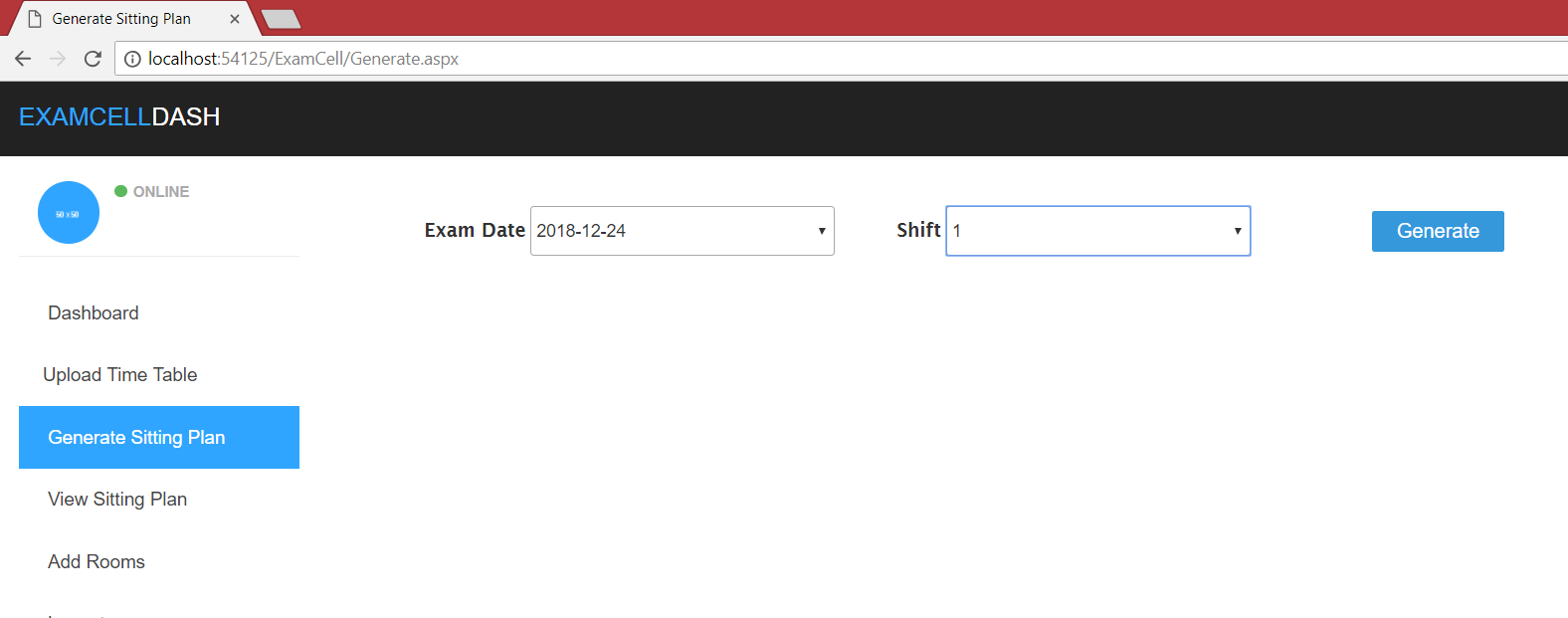
Exam Cell Dashboard

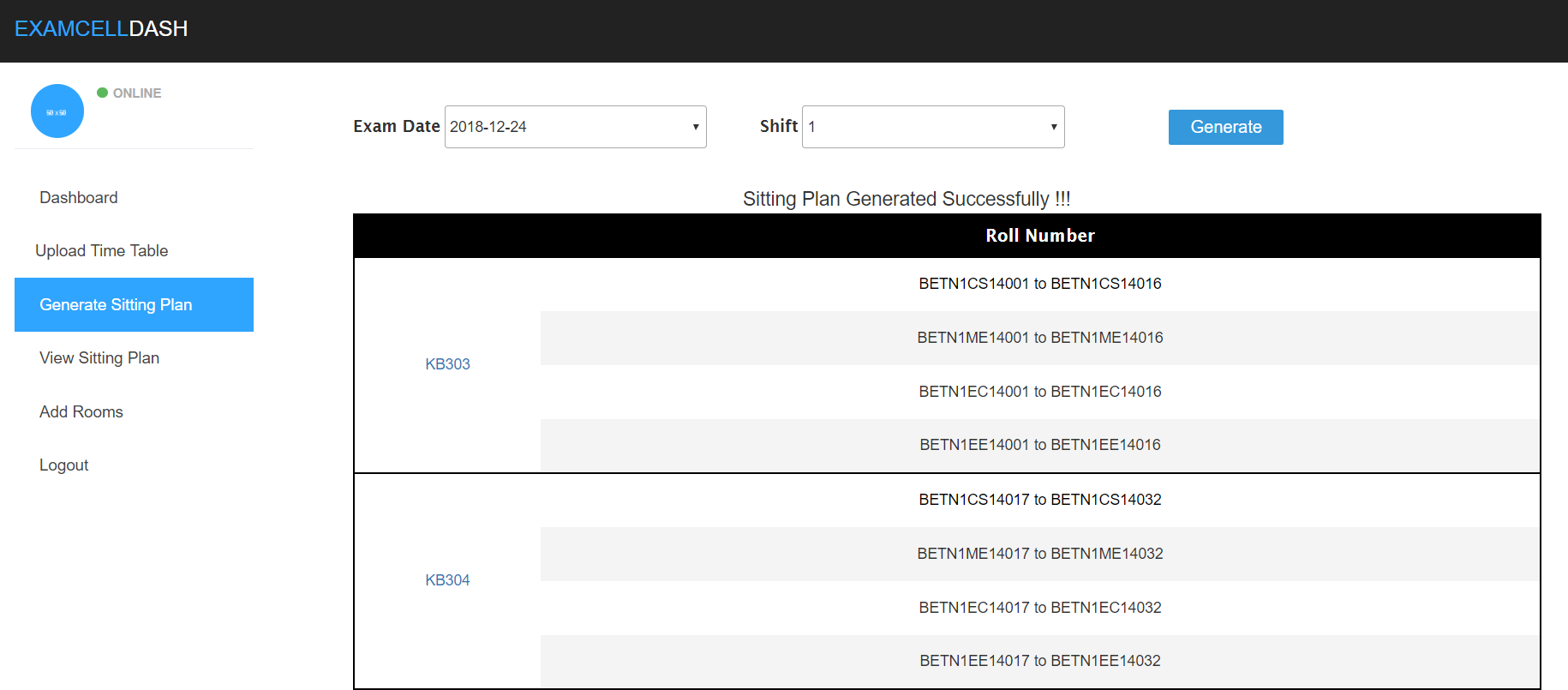


Upload Time Table

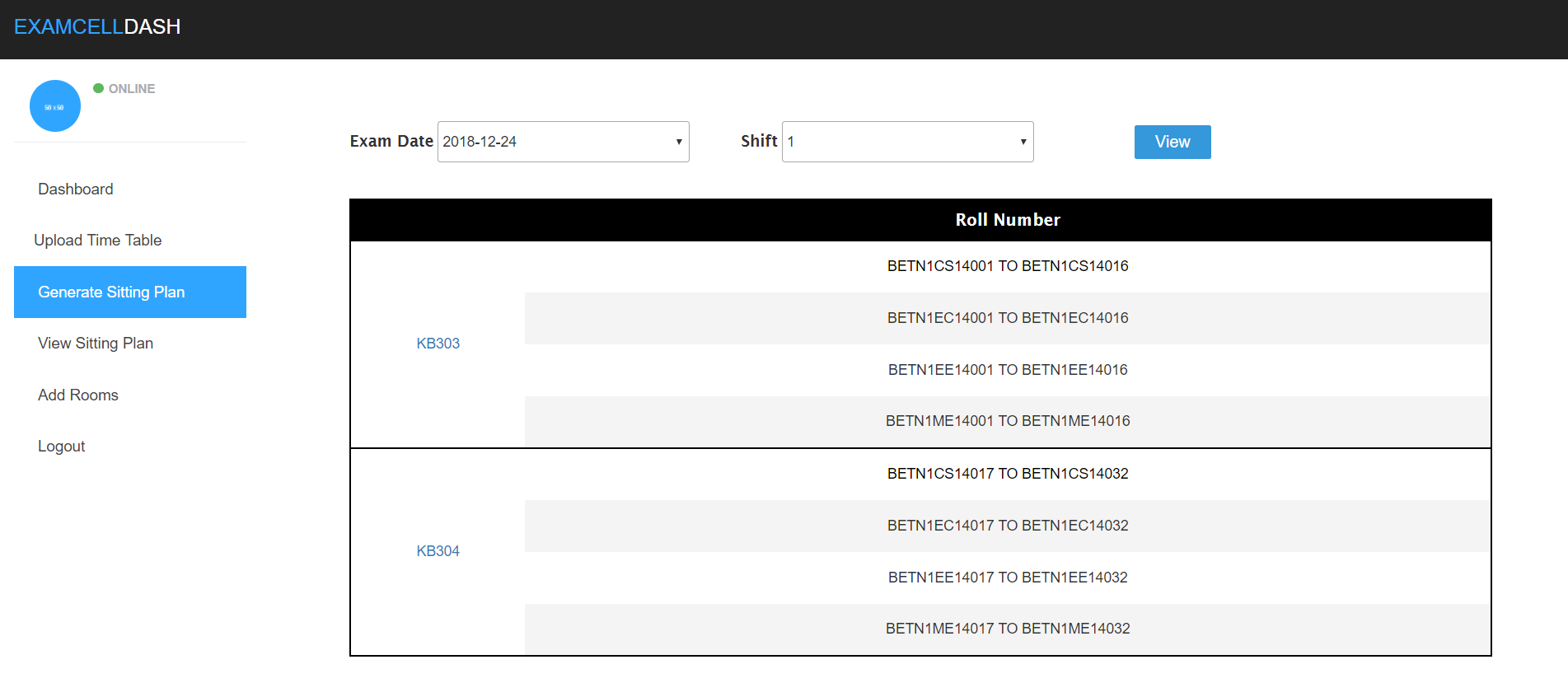


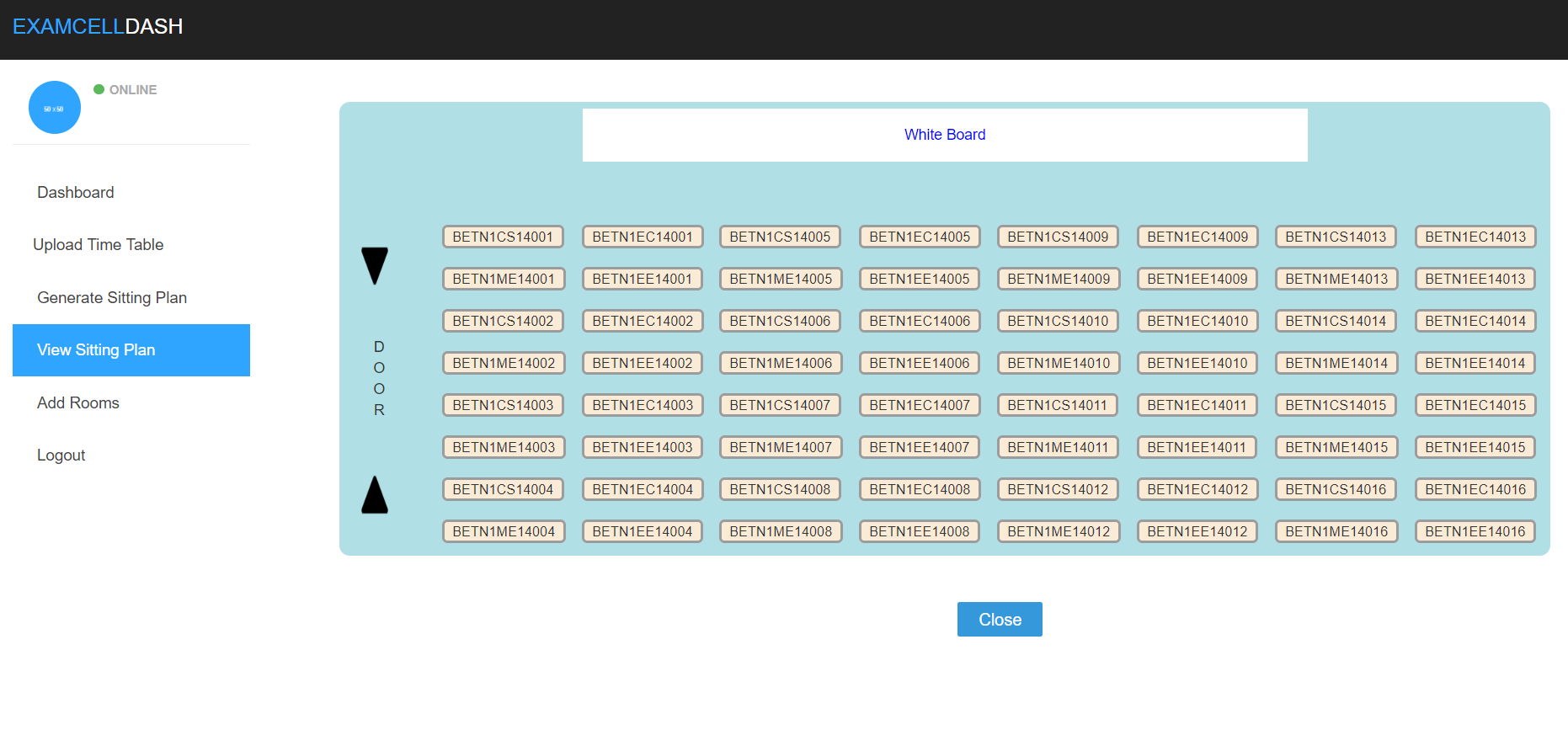
Generate Sitting Plan



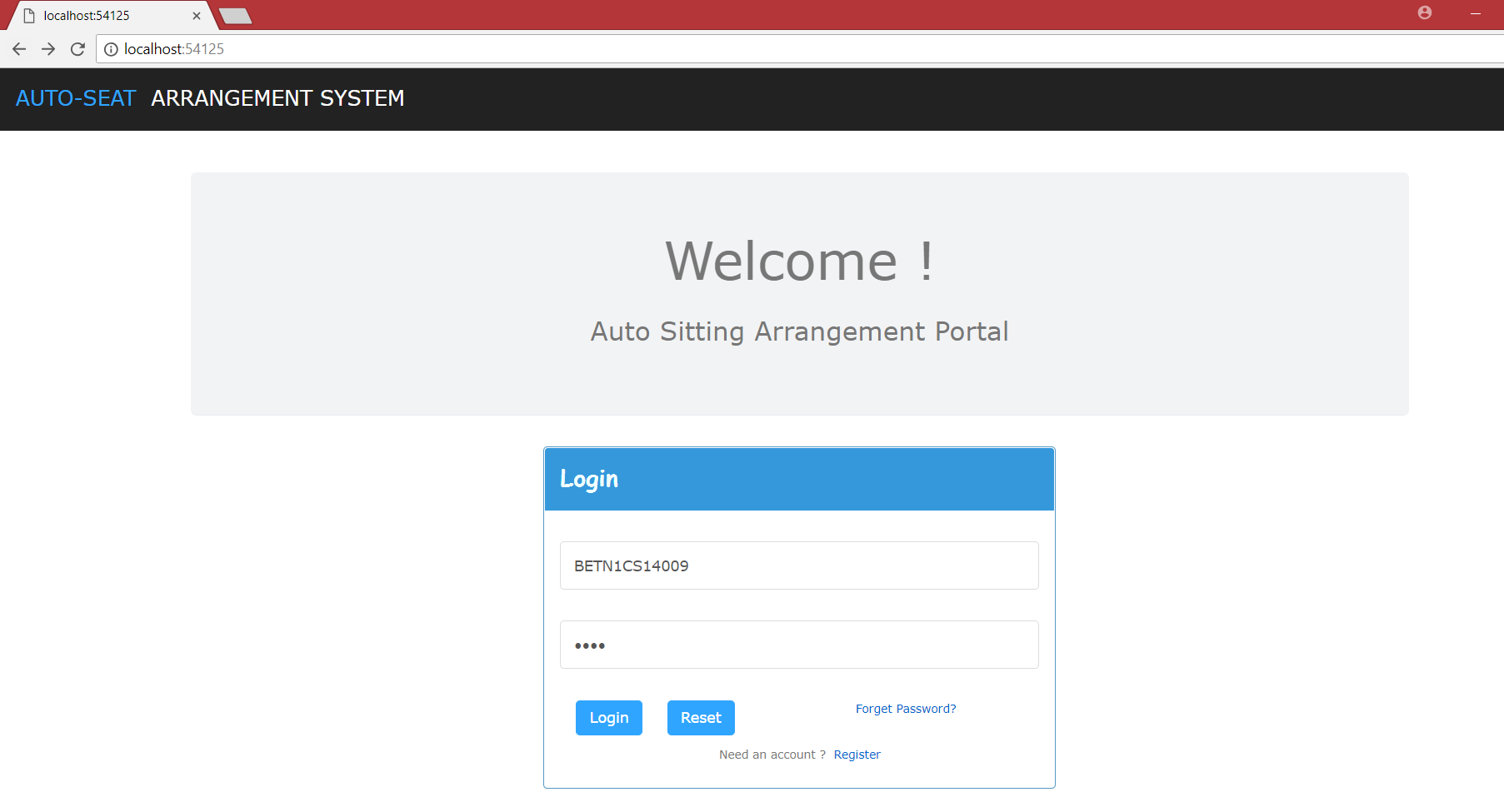


View Sitting Plan

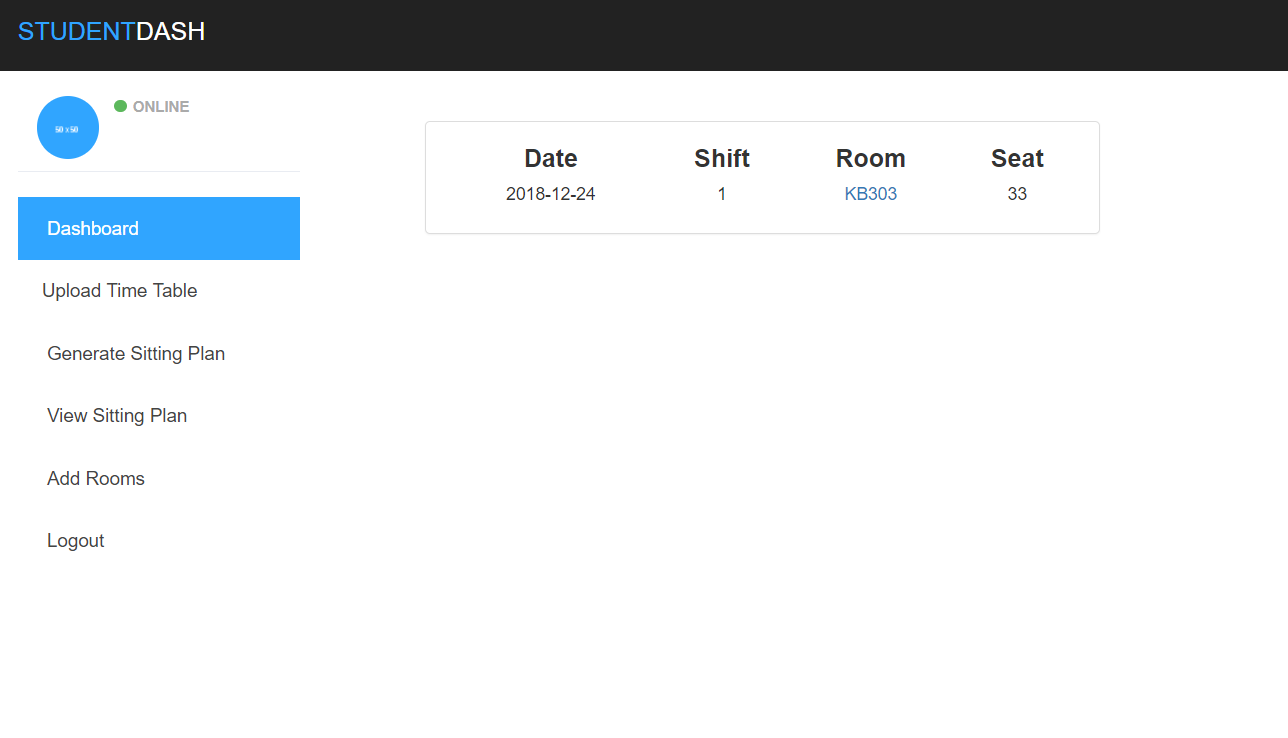




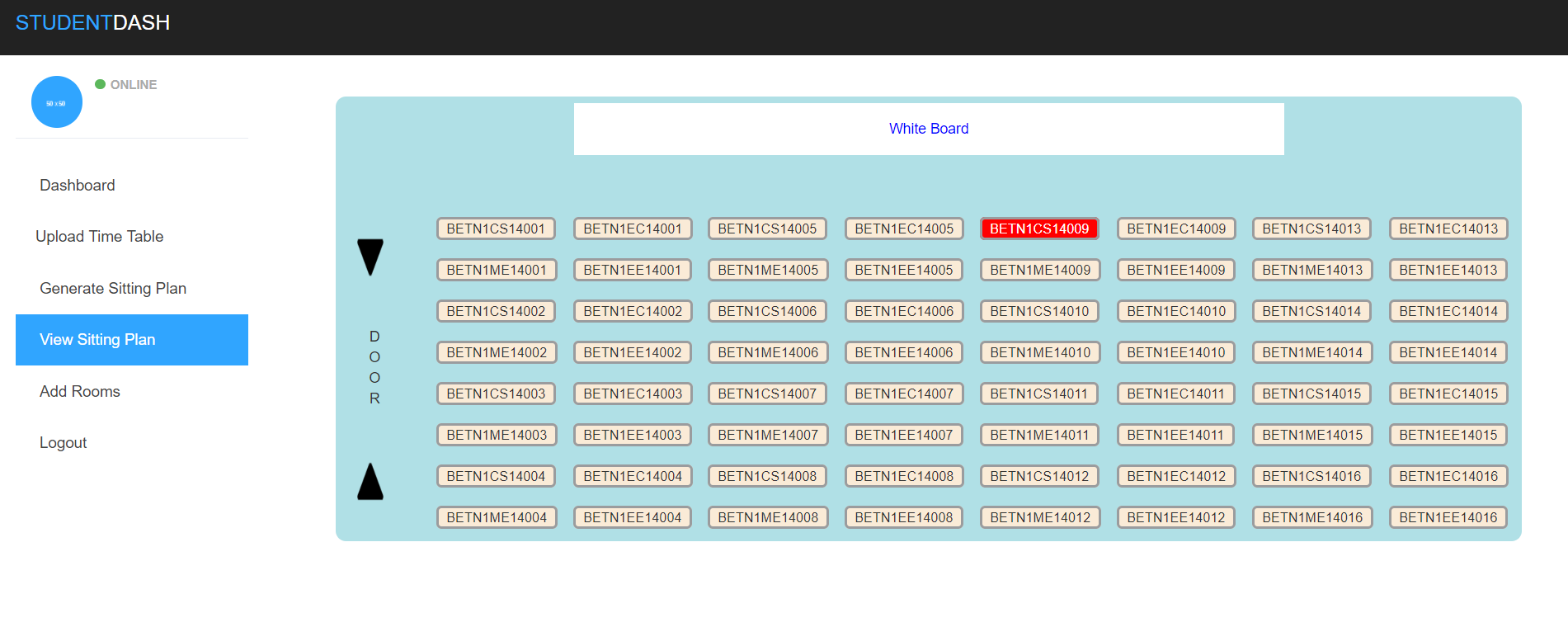
Login for student



Student Dashboard



Room view of sitting plan



**Bibliography**

<https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/>

<https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm>

<https://www.tutorialspoint.com/html/index.htm>

<https://en.wikipedia.org/wiki/HTML#History>

<https://www.wired.com/1997/04/a-brief-history-of-html/>

<http://www.yourhtmlsource.com/starthere/historyofhtml.html>

<https://www.go4expert.com/articles/html4-vs-html5-comparison-t30141/>

<https://en.wikipedia.org/wiki/Cascading_Style_Sheets#CSS_1>

<https://www.sitesbay.com/bootstrap/bootstrap-structure-of-bootstrap>

<https://www.sitesbay.com/bootstrap/bootstrap-features-of-bootstrap>

<https://www.sitesbay.com/javascript/javascript-features>

<https://en.wikipedia.org/wiki/Ajax_(programming)#Technologies>

<https://www.w3schools.com/xml/ajax_intro.asp>

<http://www.tutorialsteacher.com/csharp/csharp-version-history>

<https://www.dotnettricks.com/learn/aspnet/a-brief-version-history-of-aspnet>

<https://www.brighthub.com/internet/web-development/articles/5009.aspx>

<https://stackoverflow.com/>

<https://www.dotnetperls.com/>

<https://www.c-sharpcorner.com/>

<https://msdn.microsoft.com/en-us/>

<https://www.aspsnippets.com/>