**ABSTRACT**

In this thesis a code-editor was implemented as a part of a bigger web-based system for solving programming assignments in the course TDT4100. The editor was created in order to allow the students of the class to focus solely on writing code, and not on setting up the surrounding framework (installing programming languages and IDEs, setting up projects, etc.).

The editor supports syntax highlighting, error checking, code completion, multiples classes, and running of tests, along with all of the more basic editor functionality such as block indentation, bracket matching, line-numbers, etc. The editor is embedded into each problem contained in an assignments, which allows students to solve basic and intermediate programming challenges directly in their web-browser, without the need for any setup.

The system also utilizes several gamification elements, as described in the thesis’ preliminary study, “Gamification of Assignment Systems” (Åse, 2014). Responsive web design principles were used while implementing the system, which allows students to check their ranks and scores from any device (cellphone, tablet, laptop and similar). This was done in order to foster competition between the students, which will in turn increase motivation even further.

The results from the experiments performed indicate that the editor is well suited for use on programming assignments in courses such as TDT4100, TDT4110 and TDT4120, or any other course which has assignments that can be tested programmatically, as the editor has a low response time even for very large programs (64KB). However, the editor is not suited for courses such as TDT4180, or other GUI-programming courses, since the he editor is currently limited to displaying console output and test-results.

## **INTRODUCTION**

The purpose of this project is to modernize the practical system of college by providing easy and convenient way for performing the practical.

This project is aimed at developing a practical system which will be helpful to teacher as well as student performing the practical. It is a server based practical system capable of performing the programming in core languages (C/C++/JAVA) of computer science and engineering course.

This system is trying to solve the problem of various scenarios such as platform problem, computer problems etc. As the compiler is installed on the server, even if students don’t have compilers or IDE’s installed on their machine, they can learn and perform programs and code. This also makes all of the students to submit the code as everyone has an account of their own.

The students will then observe, analyze and perform the practical in the respective technology which will be integrated right into the browsers. The teacher can view the submitted code by the student and check it. They can also react to it or reply to it if they want.

## **Motivation**

The real motivation is the actual requirement of the student as well as teacher when they perform the practical. To reduce the extra work load by the students and teachers. There are many issues like internet connectivity, different platform compatibility; each system must be installed with the required IDE’s and many more. Our primary aim was to overcome all these issues with one system so we decided to make a server based compiler and interpreter named as CODE-IT.

## **Aim**

To implement the college practical system in easier and interesting way for students and teacher and also to overcome the common issues while performing the practical

## **Objectives**

* + - To properly designed the practical system of the college for the overall development of student.
    - To implement the hard work of the teacher in fruitful way for students in improving their practical knowledge.
    - To overcome most of the common issues while performing the practical.
    - Perform proper analysis on the student practical records individually.

## **Scope**

Here the server based practical system is developed for performing practical in an easy way. As the practical is one of the most important assets of the student life to gain practical knowledge on their core programming subjects. The system provides the facility of proper execution of practical, report error or output to the students and provides an interesting view to teacher for easy analysis of student’s practical.

In future it can be implemented successfully for the practical examination on many subjects that are based on compilers and interpreters.

## **Organization of Report**

Chapter 2 provides the literature survey showing different techniques for image sharing. Chapter 3 provides the analysis of the detailed design of the proposed system. Chapter 4 of the project gives the complete implementation details of the system. Chapter 5 provides advantages and limitations of the proposed system. Conclusion is drawn in Chapter 6 along with discussions of possible future extensions.

* 1. **Background History**

A web integrated development environment is a browser based IDE that allows for software development or web development. A web IDE can be accessed from a web browser, such as Google Chrome or Mozilla Firefox, allowing for a portable work environment. A web IDE does not usually contain all of the same features as a traditional, or desktop, IDE, although all of the basic IDE features, such as syntax highlighting, are typically present.

A web IDE, like most websites, is usually composed of two pieces: a frontend and a backend. The frontend is usually written in Javascript, using AJAX methods to communicate with the backend using a HTTP API, although in some cases, a browser extension or desktop application serves as the frontend and communicates with the backend without the need for a browser. The backend takes care of creating, saving, and opening files, as well as running any terminal commands if the IDE supports it. This setup allows for portability and continuity. The state of the IDE can be saved and reopened on another machine. This also allows for compiling or running programs to continue while the user is away.

Many web IDEs support several programming languages, while others only support a specific language. Most web IDEs allow access to a Command-line interface (CLI) that allows the user to install or run any software that is needed for development, allowing "full" control over the development environment. Open source web IDEs allow for installation on local servers or machines and can be used to give the developer more control over the development environment. IDEs are designed to maximize programmers’ productivity. They normally achieve this goal by consisting of a source code editor, a compiler and/or interpreter, built-in automation tools, and a debugger. Some modern IDEs even employ plug-in frameworks that support extension to the environment, hence meeting various needs of programmers.

The startling growing software sizes and hardware consumption (e.g. memory and CPU) of IDEs as well as their plug-ins have gradually become a headache. Moreover, programmers have to ensure that their favorite IDEs and development toolkits (e.g. JDK) are installed and properly configured in their computers before they are able to start working, which takes a substantial amount of time. Even running a properly configured IDE takes a long time to load. It seems like that IDEs will add troubles to programmers these days instead of aiding them. Therefore, some interesting solutions are emerging and can keep the hundreds of megabytes away from disk.

The basic idea is to develop a useful development tool that can be used by any thin client with a web browser. It will make programming on the move, or programming outside of the normal working environment far more convenient and easy. Also, it will provide a plug-in architecture to extend the platform to support more features. However, plug-ins here no longer needs any installation or configuration: appending their URLs to the IDE’s setting will automatically trigger additional functionality.

On the other hand, web-based IDE encourages live collaboration between team members, because its structure requires all documents and codes to be modified and saved on the servers, which will release the problems of the implicit code knowledge. The most exciting advantage of web-based IDE is that, with the new generation of smart phones and PDAs that support AJAX and Java , programming jobs can be done movably anywhere anytime.

## **Related Works**

An automated web-based grading system called Infandango was developed by Hull, Powell and Klein. The system allows students to submit Java source filesand a backend JUnit test engine compiles and executes a set of predefined tests. The outcome is then stored in a database and communicated back to the students. The Infandango system is composed of four components: a web front-end, the CoSign authentication module, the Jester JUnit tester and the PostgreSQL database. Based on the authors’ conclusion, the components are loosely coupled and they can be swapped out with other appropriate components.

Web-CAT, developed by Edwards and Perez-Quinones, also provides an automatic web-based grading system. It supports mainly Java and C++ exercises. Web-CAT exercises can be configured to require the test cases together with the source files. This is one of the well-known features of Web-CAT. Web-CAT is extensible with its plug-ins. Plug-ins can be developedto support other programming languages and they can be configured to collect more statistical data from the students’ submissions.

Pritchard’s approach with Websheet is for an instructor to setup a solution and provide the locations of the "fill-in-the-blank" areas [18]. Through a browser, students provide the fill-in and submit for evaluation. Websheet supports both Java and C++ and uses CodeMirror for its text editor. It is designed for in-class exercises or practice homework problems. After three failed attempts, the solution is made available to the user.

Deeb and Hickey developed Spinoza, which is also a web-based IDE with an automatic grading engine. With SpinozaExercises, instructors can quickly create a problem by providing a description and optional skeleton codes. SpinozaHomeworks are more comprehensive in that solution files and test cases are required so they can be used to evaluate submitted solutions. A unique feature of Spinoza is the share mode where instructors can display submitted code to the entire class. The instructor view offers instructors the current progress of the class.

In real time, the instructors can see the students that have completed the assignment, those that still have syntax errors and those that have the same

equivalent class file. A survey was done by the authors with 238 students and found that 36% preferred this system over lectures with Powerpoint. Also, 28% of the students thought Spinoza is as good as a learning method as Powerpoint lectures.

Collabode from Goldman, Little and Miller enables programmers to synchronously collaborate and immediately share the changes with one another. The web-based development environment is powered by Eclipse on the server side. Each user accesses the files via their respective browsers. The multiple editors are supported by EtherPad, an open-source collaborative online editor, which shares the changes in near real-time. The system supports the Python and Java programming languages. The authors describe three collaborative scenarios: micro-outsourcing, test-driven paired programming and mobile instructors. With micro-outsourcing, many programmers can make small contributions to a developer. And in the last scenario, instructors can help the students by connecting to their IDEs. Kurtz, Fenwick, Tashakkori, Esmaili and Tate introduced the Code Magnet microlabs into their lecture. Students use tablets during lectures to graphically compose solutions to in-class conceptual problems. The submitted solutions are evaluated by Code Magnet and feedback is provided. A control study with two different groups of students was done at SUNY Stony Brook where three identical questions were asked on their tests. The questions were taken from lectures on the binary tree traversals, the building of binary search tree (BST) from postorder traversal and the building of general binary tree from preorder and inorder traversals. The authors found that the group that had been exposed to microlabs performed, on average, better than those that had not been by 8 to 10% with a statistical significance of p = 0.038.

## **Summary & Discussion**

While there currently exists popular IDEs such as Eclipse, however, there seems not to be a huge demand for web based IDEs. We state against this belief, claiming that web based IDEs offer functionality not provided by traditional IDEs.

Aside from including features of all web based applications, such as thin clients and any time/any place services, web based IDEs have other unique features. The compiling on the server side can be done on powerful and separate processors, saving the client all the trouble. Also, the client does not need to have all SDKs installed to be able to use them.

There is another alternative for web based IDEs i.e. a remote desktop connection (RDC). For an organization, RDC allows the IT department to install applications on a central server instead of multiple computers. Remote users can log on and use those applications over the network. Such centralization can make maintenance and troubleshooting easier. RDC and Windows authentication systems prevent unauthorized users from accessing apps or data. RDC is successfully applied in multiple industries for the management of IT infrastructure.

As all these feature are also provided by a remote desktop connection (RDC), but the ability to share a project between multiple users, allowing them to develop different parts of a project simultaneously without extra coordination gives web based IDEs an edge over RDCs. However, RDC can be ineffective in transmissions and inconvenient to use. What’s more, we need to have another remote computer that we have access to and an RDC tool to make the connection happen.

## Proposed System Analysis & Design

* 1. **Problem Statement**

As in practical sessions we need to install all the related software for performing the practical on C/C++ and Java on every individual system along with this teacher have to note the student performance as per the schedule. But the problem is that no such systems are available or not in use for college practical. Therefore, there is a need of college practical systems which can assist teacher to easily and properly configure practical by uploading list for practical and complete analysis of individual students. Also there will be simple and effective practical performance by the student.

## **Analysis**

Practical is one of the most important assets of student life that help them to gain practical knowledge. Today when practical session for the core subjects of computer engineering (C/C++/Java) starts, we need to install different IDE or compilers in every individual system. Problems occur when some software for performing the practical is not compatible with the system. This modern method of performing practical leads to the end of all the above issues discussed. To make this practical system interesting and fruitful to every student, CODE-IT can be easy to implement. CODE-IT is a server based compiler means any computer system can access this practical system connected to the server and can perform practical on various subjects. Another point is that to provide an interface to teacher that can make complete analysis of each and every student. By this system teacher can easily upload the practical list with the aim of the practical along with some little description about it.

## **Feasibility Study**

Feasibility means whether some idea will work or not. In other work, feasibility study involves an examination of the operations. A project feasibility study is an exercise that involves documenting each of the potential solutions to a particular business problem or opportunity. Feasibility studies can be undertaken by any type of business, project or team and as a critical part of the project life cycle. A procedure that identifies, describes, and evaluates candidate systems and selects the best system for the job is called as Feasibility study.

Three key considerations are involved in the feasibility analysis:

### **Technical Feasibility:-**

The use of HTML, CSS, Java Script and JSP makes form design easy and convenient. The project can be run on any system with minimum requirements. It reduces data entry errors because of applying validation in most of all the forms, it can be easily handled by any user, and it also helps in faster data updations. Also the project though developed in latest GUI, it is very easy to operate. Hence the project is technically feasible.

### **Economic Feasibility:-**

Cost benefit analysis is very important in deciding whether the project is economically feasible or not. It is alone sufficient to save our time and money. It is one time investment and does not require regular maintenance. Through cost benefit analysis it was concluded that the benefits outweigh costs and thus the project is economically feasible.

### **Behavioral Feasibility:-**

Behavioral feasibility determines how much effort will go into educating, selling and training the users on a candidate system. As everyone now-a-days are users of social Network it is very easy to handle normally by anyone. The project was also evaluated to be behaviorally feasible as it is very user-friendly and hardly needs any extra efforts to educate user for its utility and functioning.

## **Requirement Analysis**

In this system to develop a college practical system we have to study the requirement and dependencies for its development. Analysis shows that, unfortunately users struggle to set up and maintain status of practical performed hence this process is little bit tedious and time consuming. This proposed practical system mainly consisting of student interface for performing practical, teacher interface for managing and configuring the students as well as practical and compiler and interpreter for performing C,C++ and java programs successfully.

## **System Design**

* + 1. **Basic Idea**

To design a college practical system such that it should be easy to implement and can be able to overcome their related issues. In this college practical system two main entities students and teachers are involved. Teachers have been given an interface where they can perform analysis on the practical performed by the individual student and can accept or reject the particular practical also teacher can upload aim of the practical with some description for their respective classes.

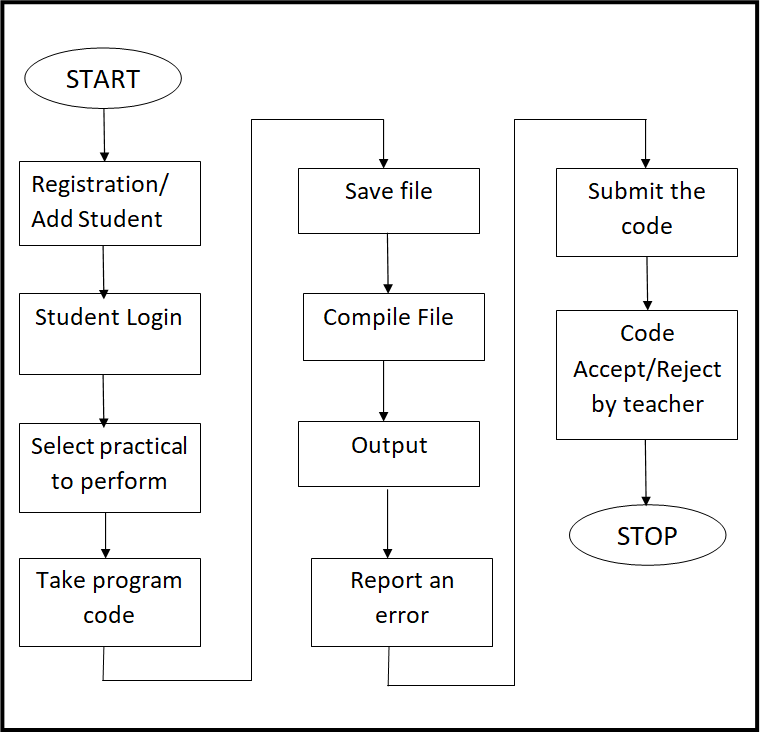
Students on the other hand have to register with their credentials and their current academic year so that they can login into the system and can perform the practical on C, C++ and Java related to them. Students need to run their program code, solve errors in their program if any and submit their program code to the system so that teacher would be able to check it and accept or reject accordingly.

To make this system in used to other than student not registered, for practicing themselves we have implemented compilers that can run the program code on C, C++ and JAVA language.

## **Algorithms Used**

* + - 1. **Step 1:** Classify the code based on the programming language.
      2. **Step 2:** Execute the respective function based the programming language.
      3. **Step 3:** Take the input and code provided and save it into file with respective extension
      4. **Step 4:** Compile the file providing inputs (if any).
      5. **Step 5:** Save the output in a variable.
      6. **Step 6:** Return the variable.
      7. **Step 7:** Display the output.

## **Working Methodology**



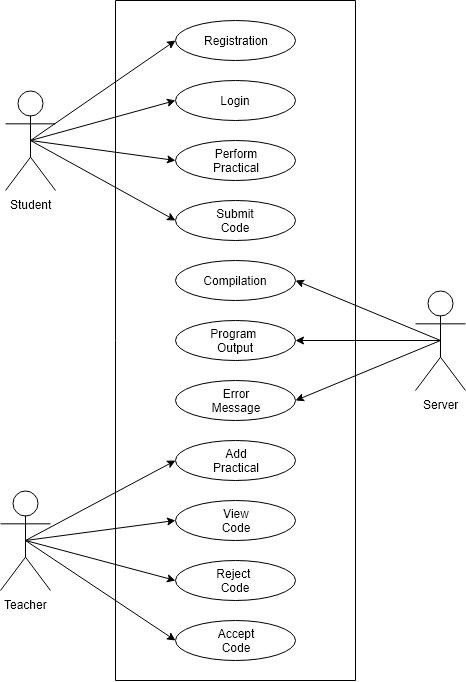
### Figure 3.3.3.1: Dataflow Diagram of Proposed System

The stepwise workflow of this working methodology using data flow diagram shown above is mention below. These steps completely describe the above working methodology.

### Stepwise Workflow of working system:

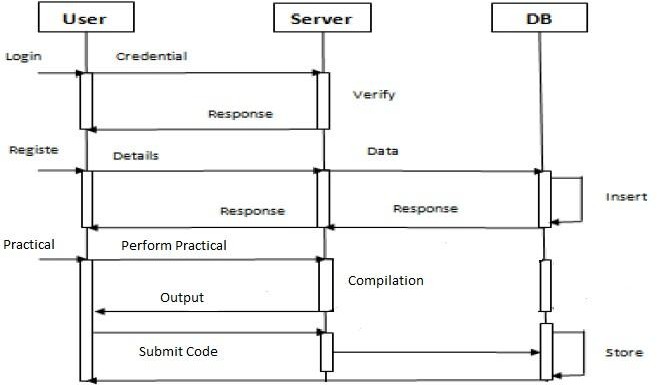
* + - 1. **Step 1:** Registration for adding new student
      2. **Step 2:** Student login to perform their respective practical.
      3. **Step 3:** Select practical to perform.
      4. **Step 3:** Take input (program code) from student.
      5. **Step 4:** Save it in a respective files (C/C++/Java).
      6. **Step 5:** Compile (Execute the file).
      7. **Step 6:** Take the output from the compilation process and send it back to the user.
      8. **Step 7:** Report any error and total compilation time.
      9. **Step 8:** Submit the program code.
      10. **Step 8:** Teacher login to Perform Analysis of student practical based on their submitted code and comments on particular practical perform.

## **Use case Diagram**

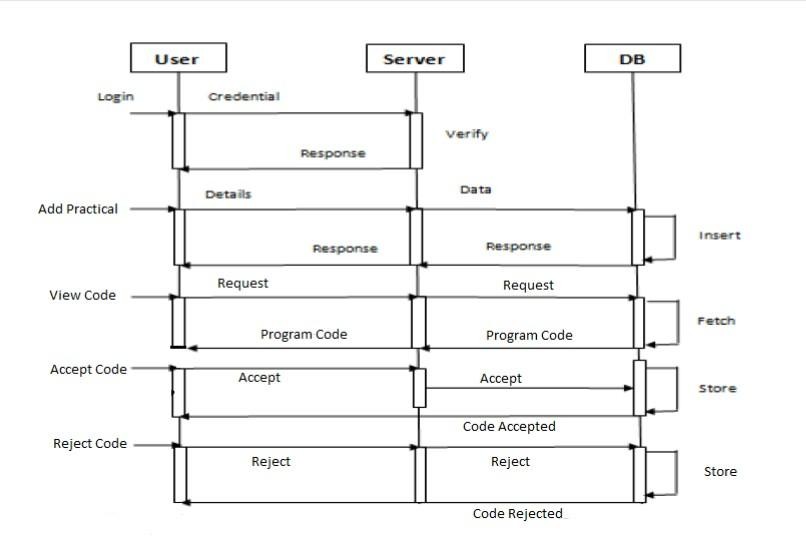


**Figure 3.3.4.1: Use Case Diagram**

## **Sequence Diagram**



### **Figure 3.3.5.1: Sequence Diagram for Student**



**Figure 3.3.5.2: Sequence Diagram for Teacher**

## Technology Implementation & Testing

* 1. **Technology**

## HTML

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 the documents 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 web page. HTML 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 /> directly introduce content into the page. Other tags 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 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. HTML can embed programs written in a scripting language such as JavaScript which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

## CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages 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 makes it possible to present the same markup page in different styles for different webpage. Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The styles can also be placed in an external CSS file, as described below, and loaded using syntax similar to:

<link href="path/to/file.css" rel="stylesheet" type="text/css">

Here link tag is used under the head tag (<head> </head>) to link the external css file with the HTML page, href is used to specify the location of the .css file, rel specify the relation between the HTML page and css document as stylesheet and type defines the type of file which we want to link.

## JavaScript

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. It is used to make dynamic web pages interactive and provide online programs, including video games. The majority of websites employ it and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming

## React js

React js is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT License. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

React js's syntax is designed to make it easier to navigate a document, select DOM elements, handle events, and develop Ajax applications. React js also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the React js library allows the creation of powerful dynamic web pages and Web applications.

## Materialize CSS

Materialize is a UI component library created with CSS, JavaScript, and HTML. Materialize UI components help in constructing attractive, consistent, and functional web pages and web apps, while adhering to modern web design principles such as browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites. It is inspired from Google

Material Design. Materialize provides you ready made rich layouts, animations and many more powerful UI.

Materialize CSS have rich set of colors available that can be easily applied to the text as well as background. There are various readymade components available that can be used for attractive and simple designing of the webpage.

Some of them are as follows:

* + - * **Cards**: - Cards are a convenient means of displaying content composed of different types of objects. They’re also well-suited for presenting similar objects whose size or supported actions can vary considerably, like photos with captions of variable length.
      * **Buttons**: - There are 3 main button types described in material design. The raised button is a standard button that signify actions and seek to give depth to a mostly flat page. The floating circular action button is meant for very important functions. Flat buttons are usually used within elements that already have depth like cards or modals.
      * **Navbar**: - The navbar is fully contained by an HTML5 Nav tag. Inside a recommended container div, there are 2 main parts of the navbar. A logo or brand link, and the navigations links. You can align these links to the left or right.
      * **Grid**: - We are using a standard 12 column fluid responsive grid system. The grid helps you layout your page in an ordered, easy fashion.
      * **Tabs**: - The tabs structure consists of an unordered list of tabs that have hashes corresponding to tab ids. Then when you click on each tab, only the container with the corresponding tab id will become visible.

## NODE JS

NODE JS is a server-side scripting language designed for web development but also used as a general-purpose programming language.

NODE JS code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. NODE JS code is usually processed by a NODE JS interpreter implemented as a module in the web server. The web server combines the results of the interpreted and executed NODE JS code, which may be any type of data, including images, with the generated web page.

Instead of lots of commands to output NODE JS pages contain HTML with embedded code that does “something” (in this case, output “This is a NODE JS Script!”). The NODE JS code is enclosed in special start and end processing instruction <?Node js and

?> that allow you to jump into and out of “NODE JS Mode.”

What distinguishes NODE JS from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to client. NODE JS is mainly focused on server side scripting which can be used for collecting form data, generating dynamic page content. The best things in using NODE JS are that it is extremely simple for beginners to start, but offers many advanced features for a professional programmer.

## **MySql**

MySQL is an open-source relational database management system (RDBMS). Its name is combinations of “My”, the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl / NODE JS / Python". MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

## Codeigniter Framework

CodeIgniter is a powerful NODE JS framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications.CodeIgniter is loosely based on the popular model–view–controller (MVC) development pattern. While controller classes are a necessary part of development under CodeIgniter, models and views are optional. Codeigniter can be also modified to use Hierarchical Model View Controller (HMVC) which allows developers to maintain modular grouping of Controller, Models and View arranged in a sub-directory format.

* + - * **Model**: - The Model represents your data structures. Typically, your model classes will contain functions that help you retrieve, insert and update information in your database.
      * **View**: - The View is information that is being presented to a user. A View will normally be a web page, but in CodeIgniter, a view can also be a page

fragment like a header or footer. It can also be an RSS page, or any other type of “page”.

* + - * **Controller**: - The Controller serves as an intermediary between the Model, the View, and any other resources needed to process the HTTP request and generate a web page.

CodeIgniter is most often noted for its speed when compared to other NODE JS frameworks. CodeIgniter is faster, lighter and the least like a framework.

## Wamp Server

WampServer refers to a software stack for the Microsoft Windows operating system, created by Romain Bourdon and consisting of the Apache web server, OpenSSL for SSL support, MySQL database and NODE JS programming language.

Wamp Stands for "Windows, Apache, MySQL, and NODE JS." WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and NODE JS). It is often used for web development and internal testing, but may also be used to serve live websites.

The most important part of the WAMP package is Apache (or "Apache HTTP Server") which is used run the web server within Windows. By running a local Apache web server on a Windows machine, a web developer can test webpages in a web browser without publishing them live on the Internet.

WAMP also includes MySQL and NODE JS, which are two of the most common technologies used for creating dynamic websites. MySQL is a high-speed database, while NODE JS is a scripting language that can be used to access data from the database. By installing these two components locally, a developer can build and test a dynamic website before publishing it to a public web server.

While Apache, MySQL, and NODE JS are open source components that can be installed individually, they are usually installed together. One popular package is called "WampServer," which provides a user-friendly way to install and configure the "AMP" components on Windows.

NOTE: The "P" in WAMP can also stand for either Perl or Python, which are other scripting languages. The Mac version of LAMP is known as MAMP.

## Compiler & Interpreter

* + 1. **GNU GCC**

The GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU toolchain and the standard compiler for most Unix-like operating systems. The Free Software Foundation (FSF) distributes GCC under the GNU General Public License (GNU GPL). GCC has played an important role in the growth of free software, as both a tool and an example.

Originally named the GNU C Compiler, when it only handled the C programming language, GCC 1.0 was released in 1987. It was extended to compile C++ in December of that year.

As well as being the official compiler of the GNU operating system, GCC has been adopted as the standard compiler by many other modern Unix-like computer operating systems, including Linux and the BSD family, although FreeBSD and macOS have moved to the LLVM system. Versions are also available for Microsoft Windows and other operating systems; GCC can compile code for Android and iOS.

* + - * **CodeBlocks**:- CodeBlock is a free, open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the GUI toolkit. Using plugin architecture, its capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards C, C++, and FORTRAN. It has a custom build system and optional Make support.

Code::Blocks is being developed for Windows, Linux, and macOS and has been ported to FreeBSD, OpenBSD and Solaris.

## Java Development Kit (JDK)

The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private java virtual machine (JVM) and a few other resources to finish the development of a Java Application. Since the introduction of the Java platform, it has been by far the

## Ace Editor

Ace (from Ajax.org Cloud9 Editor) is a standalone code editor written in JavaScript. The goal is to create a web-based code editor that matches and extends the features, usability, and performance of existing native editors such as TextMate, Vim, or Eclipse. It can be easily embedded in any web page and JavaScript application. Ace is developed as the primary editor for Cloud9 IDE and as the successor of the Mozilla Skywriter project.

Some of the features of Ace editor are as follows:-

* + - * Syntax highlighting.
      * Auto indentation and outdent.
      * An optional command line.
      * Work with large documents (handles hundreds of thousands of lines without issue).
      * Themes (TextMate themes can be imported).
      * Search and replace with regular expressions.
      * Highlight matching parentheses.
      * Toggle between soft tabs and real tabs.
      * Displays hidden characters.
      * Highlight selected word.

## Application, Advantages & Limitations

* 1. **Applications**

There are lots of applications of this new college practical system:

* + - Mainly useful for any college for performing the practical in the programming languages as C, C++, JAVA.
    - Ready to implement in any college easily.
    - All the students can perform practical efficiently.
    - Teacher can make easy analysis of every student performed practical.
    - Available to every system connected to the server.

## **Advantages**

Some of the advantages are as follows:

* + - Access from any computer connected to the server.
    - Minimal configuration needed (or only needed once).
    - Centralized workspace.
    - Easily included in a virtual development environment.
    - No need to install a lot of software locally.
    - Allows for development from inexpensive machines, such as Chromebooks, since the testing and development occurs on a separate machine (server).
    - Can be used as a desktop IDE when setup with a Web server on the local machine.

## **Limitations**

Some of the limitations of this system are:

* + - If not self hosted, possible outside security issues.
    - Possible server downtime.
    - Maintenance if self hosted
    - Most do not support smart phones or tablets well.

## **Setup guide**

### Setting up Eclipse

1. Start Eclipse
2. Set workspace to the root of project (the folder containing README.md)
3. Open "File" > "Import..."
4. Select "General" > "Existing Projects into Workspace"
5. Press "Next"
6. In the "Select root directory" field, browse to "/no.ntnu.assignmentsystem.model"
7. Make sure the project is checked
8. Click "Finish"

### Installing Java 8 plugin

1. Go to "Help" in the menu bar
2. Open "Eclipse Marketplace"
3. Search for java 8 kepler
4. Install "Java 8 support for Eclipse Kepler SR2"
5. Complete the wizard

### Setting up JRE8 in eclipse

1. Go to "Window" > "Preferences" > "Java" > "Installed JREs"
2. If jre8 is not in the list, click "Add"
3. Choose "Standard VM"
4. Set "JRE Home" to your jre8 path.
5. Click "Finish"

### Installing Maven plugin

1. Go to "Help" in the menu bar
2. Open "Eclipse Marketplace"
3. Search for maven 1.4
4. Install "Maven Integration for Eclipse (Juno or newer) 1.4"
5. Complete the wizard.

### Installing Akka dependencies into Eclipse

1. Open terminal and navigate to "/setup" folder
2. Run mvn p2:site (https://github.com/reficio/p2-maven-plugin)
3. Open Eclipse
4. Go to "Help" > "Install new Software..." in menu bar
5. Click "Add..."
6. Click "Local"
7. Navigate to "/setup/target/repository" and click "Open"
8. Click "OK"
9. Check "Maven osgi-bundles" in table view
10. Click "Next >"
11. Click "Finish"

### Generating model code

1. Navigate to "model/model.genmodel"
2. Right-click on "Model"
3. Click on "Generate Model Code"

### Create a Run configuration

1. Open "Run" > "Run Configurations" in menu bar
2. Right-click "Java Application" and select "New"

Main

1. Set name to
2. Set project to "no.ntnu.assignmentsystem.model"
3. Set main class to

Main

1. Click "Run" and confirm that it compiles

### Exporting to JAR

1. Right-click the project and select "Export"
2. Select "Java" > "Runnable JAR file"
3. Click "Next"
4. Set launch configuration to "Main"
5. Set export destination to "AssignmentModel/lib"
6. Set library handling to "Copy required libraries into a sub-folder next to the generated JAR"
7. Check "Save an ANT script"
8. Click "Finish"
9. Move the JAR-files from sub-folder to "AssignmentModel/lib"

### Set up automatic building

1. Right-click project and select "Properties"
2. Go to "Builders"
3. Click "New..."
4. Select "Ant Builder"
5. Click "OK"
6. Set name to Model Builder
7. Set buildfile to the generated ANT-file
8. Set base directory to the root folder (folder containing README.md)
9. Click "OK"

### Setting up IntelliJ

Generate IDEA-files (run activator idea in directory)

1. Start IntelliJ
2. Select "Open Project"
3. Navigate to "/AssignmentSystem"

## Conclusion and Future Work

* 1. **Conclusion**

In today’s world we require everything online so this all systems provide the best solution to these problems.

CODE-IT provides a key solution for online compilation and execution of college practicals in of C, C++ and Java programming languages for college students. CODE-IT enables students to compile and execute their programs without having to configuring their machine for C,C++ and Java program compilation. This also allows students to perform their practicals online, anytime.

The features provided in this system allow the student to login, perform the required practical and submit the code. And CODE-IT makes it much easier for the students to perform the practical and for the teachers to check the progress of their students

It also generates detailed statistics of student’s compilations that can help teachers to improve their teaching methodologies. Teachers can also reject student’s code if they find the logic incorrect or improper output.

## **Future Work**

There is always space for improvement in every project. This project can be further developed and additional features can be added like debugging the code, providing practice problems related to c, cpp and java programming. Initially this website includes three programming languages i.e. C, C++ and JAVA. We can improve this system by adding some more programming languages in future. We are also planning to develop this system into a mobile application for android and ios devices. So that it can reach to a much larger audience.

## **References**

1. F.A. Deeb and T. Hickey, The Spinoza code tutor: faculty poster abstract,

Journal of Computing Sciences in Colleges, 30(6):154–155, 2015

1. S.H. Edwards, Work-in-Progress: Program Grading and Feedback Generation with Web-CAT, Proceedings of the first ACM conference on Learning @ scale conference, 215–216
2. S.H. Edwards and M.A. Perez-Quinones, Web-CAT: Automatically Grading Programming, Assignments , Proceedings of the 13th annual conference on Innovation and technology in computer science education, 328–328
3. M. Goldman, G. Little and R.C. Miller, Collabode: collaborative coding in the browser, Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering, 65–68
4. M.J. Hull, D. Powell and E. Klein, Infandango: automated grading for student programming, Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, 330–330
5. D. Pritchard, Websheets: A Templated Online Coding Exercise System, Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, 335–335
6. W. Toll, Top 48 Integrated Developer Environments (IDEs) & Code Editors, https://blog.profitbricks.com/top-integrated-developerenvironments-ides/
7. Wikipedia, Comparison of integrated development environments, https://en.wikipedia.org/wiki/Comparison\_of\_integrated\_development\_environment
8. Berners-Lee, Tim, Connolly, Daniel: HyperText Markup Language Specification

– 2.0 INTERNET DRAFT. IETF. Retrieved 24 October 2010.

1. Ace Text Editor, https://ace.c9.io/#nav=about
2. Berners-Lee, Tim; Fischetti, Mark (2000). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. San Francisco: Harper. ISBN 978-0-06-251587-2.
3. Holzschlag, Molly E (2005). Spring into HTML and CSS. Pearson Education, Inc. ISBN 0-13-185586-7.
4. Burns, Joe; Growney, Andree S. (2001). *JavaScript Goodies.* Pearson Education. ISBN 0-7897-2612-2.
5. Powell, Thomas A.; Schneider, Fritz (2001). JavaScript: The Complete Reference. McGraw-Hill Companies. ISBN 0-07-219127-9.
6. Wikipedia, PHP: Hypertext Preprocessor, https://en.wikipedia.org/wiki/PHP.
7. Susan Perschke (23 April 2012). "WampServer delivers a smart, Windows- friendly platform for Apache, MySQL and PHP-based apps". Network World. Retrieved 20 January 2015.
8. "MySQL | the World's Most Popular Open-Source Database".

[19] "Which Should I Use: MySQL Enterprise or MySQL Community Server?"MySQL AB. Retrieved 8 April 2009.

**ASSESSMENT**

**Internal:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL NO** | **RUBRICS** | **FULL MARK** | **MARKS OBTAINED** | **REMARK S** |
| 1 | Understanding the relevance, scope and dimension of the project | 10 |  |  |
| 2 | Methodology | 10 |  |  |
| 3 | Quality of Analysis and Results | 10 |  |  |
| 4 | Interpretations and Conclusions | 10 |  |  |
| 5 | Report | 10 |  |  |
|  | **Total** | **50** |  |  |

**Date: Signature of the Faculty**

**COURSE OUTCOME (COs) ATTAINMENT**

* **Expected Course Outcomes (COs):**

**(Refer to COs Statement in the Syllabus)**

* **Course Outcome Attained:**

**How would you rate your learning of the subject based on the specified COs?**



**1 2 3 4 5 6 7 8 9 10**

**LOW HIGH**

* **Learning Gap (if any):**
* **Books / Manuals Referred:**

**Date: Signature of the Student**

* **Suggestions / Recommendations:**

**(By the Course Faculty)**

**Date: Signature of the Faculty**

1. [- Introduction 1](#_bookmark0)

**Table of Contents**

* 1. [Background and motivation](#_bookmark1) 2
  2. [Goals 3](#_bookmark6)
  3. [Research approach](#_bookmark7) 4
  4. [IDE functionality](#_bookmark15) 5
     1. [Syntax highlighting](#_bookmark16) 6
     2. [Error checking](#_bookmark18) 7
     3. [Code completion 9](#_bookmark20)
     4. [IDE architecture 10](#_bookmark22)
  5. [Editor performance 13](#_bookmark27)
     1. [Response time 1](#_bookmark28)4
     2. [Scalability 1](#_bookmark29)5
  6. [Responsive design 1](#_bookmark30)5
  7. [Gamification 16](#_bookmark32)
  8. [State of the art 16](#_bookmark33)
     1. [Educational systems 16](#_bookmark34)
     2. [Pure editors 19](#_bookmark38)
     3. [Comments and conclusion 2](#_bookmark44)0

1. [- Editor architecture](#_bookmark61) 22
   1. [Supported functionality](#_bookmark62) 23
   2. [The evolution of the architecture](#_bookmark64) 25
      1. [High-level architecture 2](#_bookmark65)6
      2. [Making the web-server explicit](#_bookmark69) 27
      3. [Adding core functionality](#_bookmark72) 28
      4. [Alternative architecture](#_bookmark76) 29
      5. [Conclusion 30](#_bookmark80)
   3. [Communication](#_bookmark81) 32
      1. [Communication between client and server](#_bookmark82) 33
      2. [Internal communication](#_bookmark83) 34
      3. [Editor API](#_bookmark85) 36
      4. [Conclusion](#_bookmark87) 36
   4. [Possible improvements](#_bookmark89) 37
      1. [Re-use Eclipse processes](#_bookmark90) 38
      2. [Reduce the footprint of the Eclipse software](#_bookmark94) 39
      3. [Scalability](#_bookmark95) 40
      4. [Simplify deployment](#_bookmark97) 41
2. [- Experiments and results](#_bookmark98) 42
   1. [Experimental plan](#_bookmark99) 42
   2. [Testing editor response time](#_bookmark100) 43
      1. [Startup time](#_bookmark103) 44
      2. [Running code, running tests, error checking, and code completion](#_bookmark106) 45
      3. [Conclusion](#_bookmark112) 46
   3. [Testing editor scalability](#_bookmark114) 46
      1. [Startup time](#_bookmark115) 47
      2. [Running code, running tests, error checking and code completion](#_bookmark117) 48
      3. [Conclusion 49](#_bookmark125)
3. [- Evaluation and conclusion](#_bookmark139) 50
   1. [Evaluation](#_bookmark140) 52
   2. [Discussion](#_bookmark141) 53
   3. [Future work](#_bookmark142) 54
   4. [Closing remarks](#_bookmark143) 54
4. [- Appendix A – Constraints, requirements and architecture](#_bookmark145) 55
   1. [Implementation constraints](#_bookmark146) 55
   2. [System requirements](#_bookmark147) 55
      1. [Data model](#_bookmark148) 56
      2. [Web-based code-editor](#_bookmark150) 57
      3. [Non-functional requirements](#_bookmark151) 57
      4. [Additional requirements](#_bookmark152) 57
      5. [Requirements from the preliminary study that were cut](#_bookmark153) 58
   3. [System architecture](#_bookmark154) 58
      1. [Architectural patterns](#_bookmark155) 58
      2. [System architecture in ArchiMate](#_bookmark159) 59
   4. [JavaScript frameworks and libraries](#_bookmark184) 60
      1. [Why does this project need JavaScript?](#_bookmark185) 60
      2. [Why do we need a frameworks or a library?](#_bookmark188) 61
      3. [Determining possible framework candidates 61](#_bookmark189)
      4. [Determining possible library solutions](#_bookmark192) 62
      5. [Conclusion](#_bookmark194) 63
   5. [JavaScript code-editors](#_bookmark195) 64
      1. [Why does this project need a JavaScript code-editor?](#_bookmark196) 64
      2. [Determining possible candidates](#_bookmark197) 65
      3. [Determining the best candidate](#_bookmark199) 65
      4. [Conclusion](#_bookmark201) 65
   6. [CSS-framework](#_bookmark202) 65
      1. [Why does this project need a CSS-framework?](#_bookmark203) 66
      2. [Determining possible candidates](#_bookmark204) 67
      3. [Conclusion](#_bookmark206) 67
   7. [CSS-preprocessor](#_bookmark207) 68
      1. [Why does this project need a CSS-preprocessor?](#_bookmark208) 68
      2. [Determining possible candidates](#_bookmark209) 69
      3. [LESS or SASS?](#_bookmark210) 69
      4. [Conclusion](#_bookmark214) 69
   8. [The complete stack](#_bookmark215) 70
      1. [Technology in relation to the architecture](#_bookmark218) 70
   9. [Minor technologies](#_bookmark221) 70

9. [Experiment results](#_bookmark231) 72

* + 1. [Run tests](#_bookmark242) 73
    2. [Error checking](#_bookmark247) 73
    3. [Code completion](#_bookmark252) 73
  1. [Setup guide](#_bookmark257) 74
     1. [Setting up Eclipse](#_bookmark258) 74
     2. [Installing Java 8 plugin](#_bookmark259) 75
     3. [Installing Maven plugin](#_bookmark261) 75
     4. [Installing Akka dependencies into Eclipse](#_bookmark262) 76
     5. [Set up automatic building](#_bookmark266) 77
     6. [Setting up IntelliJ](#_bookmark267) 77