

CHAPTER 1

INTRODUCTION

Time table scheduling has been in human requirements since they thought of managing time effectively. It is widely used in schools, colleges and other fields of teaching and working like crash courses, coaching centers, training programs etc. In early days, time table scheduling was done manually with a single person or some group involved in task of scheduling it, which takes lot of effort and time. While scheduling even the smallest constraints can take a lot of time and the case is even worse when the number of constraints or the amount of data to deal with increases. In such cases perfectly designed time table is reused for whole generation without any changes, proving to be dull in such situations. Other cases that can cause problem is when the number of employers/workers are weak, resulting in rescheduling of time table or they need to fill on empty seats urgently.

Institutions/Schools/Colleges/Universities are the regular users of such time tables. They need to schedule their course to meet the need of current duration and facilities that are available to them. However, their schedule should meet the requirements of new course addition and newly enrolled students to fresh batches. This may result in rescheduling the entire time table once again for its entire batches and to be scheduled in shortest possible time before the batches course start. Another problem that occurs is when scheduling time table for exams. When multiple batches have exam

on the same day, they need to be scheduled effectively taking into account all problems related to facilities that are available to conduct these exams simultaneous.

1.1 Purpose

Planning timetables is one of the most complex and error-prone applications. There are still serious problems like generation of high cost time tables are occurring while scheduling and these problems are repeating frequently. Therefore there is a great requirement for an application distributing the course evenly and without collisions. Our aim here is to develop a simple, easily understandable, efficient and portable application, which could automatically generate good quality time tables with **just-a-click** within seconds.

1.2 Benefits

TiBle, our web application allows users to generate time table for newly occurring changes in less time, with less effort and with more efficiency. It will allow users to work on and view time tables and view different information simultaneously.

1.3 Constraints types

A timetable encounters two types of constraints, hard constraints and soft constraints. Hard and soft constraints are useful to define what is an

optimized timetable or an incompatible timetable.

The constraints considered as hard are the constraints that must be satisfied.

These constraints cannot be violated. Timetable management has a lot of hard constraints. These constraints are human constraints. They are fixed and the timetables which don't respect it are not a correct solution.

A soft constraint is a constraint which may be violated. It should be satisfied for the best but the timetable is correct even if these constraints are not satisfied. These constraints can be breached.

1.4 Similar Products

- _ aSa Timetable
- _ Mimsa Small School
- _ Timetable Mate

1.5 Timeframe Schedule

The idea of this project was to create a timetable management tool suitable to the needs of students and lecturers. The first part of this project was to ask people what they exactly expect from the system, and analyze user needs. It can be done by conducting a survey, for lecturers and students prepared using Moodle tools or google documents tools for example. This is the more efficient way to do it, because the survey will cover more people than by going to door asking people.

The second step of the project was proper preparation before designing the application. It's an important part of the application to do schematics and

preparation. It was needed to know exactly what the application needed, the database design and after that, the implementation.

The developing part was the longest part, not the hardest one, but the longest. The application has been designed with PHP5 language with a design pattern to keep a code able to be reused and expanded, using a MySQL database, and JavaScript for user interaction. The written code can be documented to help next users to continue the project.

The application is able to do the following tasks successfully: A Lecturer will be able to see any semester and any section's timetable using a drop down list. Students will be able to learn more about their course details. They will be able to see and check hours to avoid conflicts. The system will generate a colored view of a timetable. Lecturers will be informed about the changes in their timetables on their e-mails. It offers an easy way to manage the profiles, the timetable tools and other information in one.

Last part was the testing and documents writing part of the project. The testing part is one of the most important. In fact, a group of teachers were assign to test the application. The idea of this part was to try to simulate a situation and see how the test-group will react, to get if the application answers to the initial survey.

The final report creates a link between the initial survey and the test of the application by the test-group composed by students and teachers.

CHAPTER 2

LITERATURE REVIEW

This chapter provides an analysis of the automated timetable literature broadly organized by algorithmic technique. It begins with a presentation of the major timetable solution generation algorithms that have persisted in the literature. A detailed examination of the academic literature is provided within the context of these fundamental solution generation algorithms. An analysis of the literature, grouped by the solution generation technique used, is then presented.

2.1 The General View

As mentioned before, many types of timetabling problems exist. But all these problems have several properties in common. One of these similarities is that certain entities have to be scheduled. For example, the high school timetabling problem has several entities such as classes or students, teachers, subjects, lessons and rooms. All these entities have properties. For example classes are linked to the subject the students of this class are taught.

Usually, these entities are differentiated into resources and events (or sometimes called meetings). In addition, constraints have to be considered. In the employee timetabling case, for instance, we find those entities, too. There are employees with different qualifications and monthly target hours or there are shifts to be assigned to employees. As already mentioned, some

of these entities are linked with others. There exist links from the shifts to the employees assigned to these shifts or from the classes to their teachers. Some of these links are fixed, such as the links from the shifts to the employees with the qualifications required to work on these shifts, and cannot be changed. Others have to be assigned during a planning process, e.g. linking a lesson to a suitable room. A planning algorithm has to construct a timetable, so we have to define what a timetable consists of. A timetable can be interpreted as an arbitrary sequence of events. To every event a certain number of time intervals are assigned, each having a starting and an ending point.

2.2 The Object-Oriented View

The above section can be used to describe timetabling problems in an object-oriented manner: There are different resources whose instances have references to each other, e.g. an instance of the subject class refers to instances of the teacher class who are able to teach that subject. Moreover, there are entities with a certain property, called events. This property is a certain time interval (or several time intervals) that is assigned to these events.

2.3 Linear Programming/Integer Programming

The Linear and Integer Programming techniques, the first applied to timetable, were developed from the broader area of mathematical programming. Mathematical programming is applicable to the class of problems characterized by a large number of variables that intersect within

boundaries imposed by a set of restraining conditions (Thompson, 1967). The word "programming" means planning in this context and is related to the type of application (Feiring, 1986). This scheme of programming was developed during World War II in connection with finding optimal strategies for conducting the war effort and used afterwards in the fields of industry, commerce and government services (Bunday, 1984).

Linear Programming (LP) is that subset of mathematical programming concerned with the efficient allocation of limited resources to known activities with the objective of meeting a desired goal such as maximizing profits or minimizing costs (Feiring, 1986).

Integer Programming (IP) deals with the solution of mathematical programming problems in which some or all of the variables can assume non-negative integer values only. Although LP methods are very valuable in formulating and solving problems related to the efficient use of limited resources they are not restricted to only these problems (Bunday, 1984).

Linear programming problems are generally acknowledged to be efficiently solved by just three methods, namely the graphical method, the simplex method, and the transportation method (see e.g., Palmers and Innes, 1976; Makower and Williamson, 1985).

The construction of a linear programming model involves three successive problem-solving steps. The first step identifies the unknown or independent decision variables. Step two requires the identification of the constraints and the formulation of these constraints as linear equations. Finally, in step

three, the objective function is identified and written as a linear function of the decision variables.

It is demonstrated that the literature is currently converging on the use of constraint based solution algorithms and implementations. It is also noted that the next most commonly reported implementation involves the use of hybrid algorithms.

2.4 Rich Internet Applications

Rich Internet applications (RIA) are web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e., maintaining the state of the program, the data etc.) back on the application server. We have used the Google Web Toolkit, which is RIA framework, for the same purpose.

Our project reduces the overhead on server of rendering client's UI components and makes room for processing time of Timetable Generator Algorithm.

In our project, timetabling problem is formulated as a constraint satisfaction problem and we proposed a practical timetabling algorithm which is capable of taking care of both strong and weak constraints and finding variables' instantiation, which is based on the forward tracking search.

During designing of our project, we first decided to define the way of representation of Knowledgebase. Since representation of knowledge base was certainly going to affect our way of thinking during building of

timetabling algorithm. We decided that XML is appropriate because of its inherent features which are discussed in next section.

Our knowledgebase is in the middle, because it is between our timetabling algorithm and GUI front end which is designed in the last. After the representation of KB is standardized, we designed the timetabling algorithm. The design of timetabling algorithm took most of our total time. During design of algorithm, first problem was, from where to start? Second problem was, does it really going to work? But after all due to our superior design of knowledgebase, flowcharts and enough thinking on timetabling data structure representation helped us to really boosted building our fine working algorithm.

The algorithm which we have built totally uses our knowledge of theory subjects including programming methodology, data structure and designing and analysis of algorithms. The previous efforts which we have spent on learning these subjects at that time helped us during designing and building of our whole project, especially in our timetabling algorithm.

After completion of testing of our timetable generator algorithm, we moved towards designing of Front End. Our prime aim was to build the Rich Internet Application (RIA). Why we have chosen RIA architecture is explained in details in next section along with our survey for various RIA architectures. Once the survey was over we choose the Google's Web Toolkit (GWT) for building RIA. This decision is taken during our Requirement and Analysis phase and details GWT is explained in that section.

CHAPTER 3

Theoretical Framework

3.1 About PHP

3.1.1 HISTORY OF PHP

PHP was conceived sometime in the fall of 1994 by Rasmus Lerdorf. Early non-released versions were used on his home page to keep track of who was looking at his online resume. The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools. It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff. The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

It is difficult to give any hard statistics, but it is estimated that by late 1996 PHP/FI was in use on at least 15,000 web sites around the world. Mid-1997 also saw a change in the development of PHP. It changed from being Rasmus' own pet project that a handful of people had contributed to, to being a much more organized team effort. The parser was rewritten from scratch by Zeev Suraski and Andi Gutmans and this new parser formed the basis for PHP Version 3.

A lot of the utility code from PHP/FI was ported over to PHP3 and a lot of it was completely rewritten. Today (end-1999) either PHP/FI or PHP3 ships with a number of commercial products such as C2's StrongHold web server and RedHat Linux. A conservative estimate based on an extrapolation from numbers provided by NetCraft would be that PHP is in use on over 1,000,000 sites around the world. To put that in perspective, that is more sites than run Netscape's flagship Enterprise server on the Internet.

3.1.2 Overview of PHP

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Instead of lots of commands to output HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does "something". The PHP code is enclosed in special start and end processing instructions `<?php` and `?>` that allow you to jump into and out of "PHP mode."

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer,

but offers many advanced features for a professional programmer. Although PHP's development is focused on server-side scripting, you can do much more with it.

3.1.3 FEATURES OF PHP

- HTTP authentication with PHP
- Cookies
- Sessions
- Dealing with XForms
- Handling file uploads
- POST method uploads
- Error Messages Explained
- Common Pitfalls
- Uploading multiple files
- PUT method support
- Using remote files
- Connection handling
- Persistent Database Connections
- Safe Mode
- Security and Safe Mode
- Functions restricted/disabled by safe mode
- Command line usage — Using PHP from the command line
- Introduction
- Differences to other SAPIs
- Options — Command line options

- Usage — Executing PHP files
- I/O streams — Input/output streams
- Interactive shell
- Built-in web server
- INI settings
- Garbage Collection
- Reference Counting Basics
- Collecting Cycles
- Performance Considerations

3.2 About JavaScript

3.2.1 Overview

Java script is a general purpose, prototype based, object oriented scripting language developed jointly by sun and Netscape and is meant for the WWW.

Java script borrows most of its syntax from java but also inherits from awk and perl, with some indirect influence from self in its object prototype system.

Java Script is almost as easy to learn as HTML and it can be included directly in HTML documents. Java Script was developed independently of java. Java script is a high level scripting language that does not depend on or expose particular machine representations or operating system services.

3.2.2 FEATURES:

- Java script is embedded into HTML documents and is executed with in them.
- Java script is browser dependent.
- JavaScript is an interpreted language that can be interpreted by the browser at run time.
- Java script is loosely typed language
- Java script is an object-based language.
- Java script is an Event-Driven language and supports event handlers to specify the functionality of a button.

3.3 About Rich Internet Applications

3.3.1 Introduction

Rich Internet applications (RIA) are web applications that have the features and functionalities of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e., maintaining the state of the program, the data etc.) back on the application server.

RIAs typically:

- run in a web browser, or do not require software installation
- run locally in a secure environment called a sandbox.

3.3.2History

The term "Rich Internet Application" was introduced in a white paper of March 2002 by Macromedia, though the concept had existed for a number of years earlier under names such as:

- Remote Scripting, by Microsoft, circa 1998
- X Internet, by Forrester Research in October 2000
- Rich (web) clients
- Rich web application

Traditional web applications centered all activity around a client-server architecture with a thin client. Under this system all processing is done on the server, and the client is only used to display static (in this case HTML) contents. The biggest drawback with this system is that all interaction with the application must pass through the server, which requires data to be sent to the server, the server to respond, and the page to be reloaded on the client with the response. By using a client side technology which can execute instructions on the client's computer, RIAs can circumvent this slow and synchronous loop for many user interactions.

This difference is somewhat analogous to the difference between "terminal and mainframe" and Client-server/Fat client approaches.

Internet standards have evolved slowly and continually over time to accommodate these techniques, so it is hard to draw a strict line between what constitutes an RIA and what doesn't. But all RIAs share one characteristic that they introduce an intermediate layer of code, often called a client engine, between the user and the server. This client engine is usually downloaded at the beginning of the application, and may be supplemented by further code downloads as the application progresses. The

client engine acts as an extension of the browser, and usually takes over responsibility for rendering the application's user interface and for server communication.

What can be done in an RIA may be limited by the capabilities of the system used on the client. But in general, the client engine is programmed to perform application functions that its designer believes will enhance some aspect of the user interface, or improve its responsiveness when handling certain user interactions, compared to a standard Web browser implementation. Also, while simply adding a client engine does not force an application to depart from the normal synchronous pattern of interactions between browser and server, in most RIAs the client engine performs additional asynchronous communications with servers.

3.3.3 General Benefits

Although developing applications to run in a web browser is a much more limiting, difficult, and intricate process than developing a regular desktop application, the efforts are often justified because:

1. Installation is not required -- updating and distributing the application is an instant, automatically handled process
2. updates/upgrades to new versions are automatic
3. users can use the application from any computer with an internet connection, and usually regardless of what operating system that computer is running

4. web-based applications are generally less prone to viral infection than running an actual executable

Because RIAs employ a client engine to interact with the user, they are:

1. **Richer.** They can offer user-interface behaviors not obtainable using only the HTML widgets available to standard browser-based Web applications. This richer functionality may include anything that can be implemented in the technology being used on the client side, including drag and drop, using a slider to change data, calculations performed only by the client and which do not need to be sent back to the server, for example, a mortgage calculator.

2. **More responsive.** The interface behaviors are typically much more responsive than those of a standard Web browser that must always interact with a remote server.

3.3.4 Performance Benefits

The most sophisticated examples of RIAs exhibit a look and feel approaching that of a desktop environment. Using a client engine can also produce other performance benefits:

1. **Client/Server balance.** The demand for client and server computing resources is better balanced, so that the Web server need not be the workhorse that it is with a traditional Web application. This frees

server resources, allowing the same server hardware to handle more client sessions concurrently.

2. **Asynchronous communication.** The client engine can interact with the server without waiting for the user to perform an interface action such as clicking on a button or link. This allows the user to view and interact with the page asynchronously from the client engine's communication with the server. This option allows RIA designers to move data between the client and the server without making the user wait. Perhaps the most common application of this is prefetching, in which an application anticipates a future need for certain data, and downloads it to the client before the user requests it, thereby speeding up a subsequent response. Google Maps uses this technique to move adjacent map segments to the client before the user scrolls their view.
3. **Network efficiency.** The network traffic may also be significantly reduced because an application-specific client engine can be more intelligent than a standard Web browser when deciding what data needs to be exchanged with servers. This can speed up individual requests or responses because less data is being transferred for each interaction, and overall network load is reduced. However, use of asynchronous pre fetching techniques can neutralize or even reverse this potential benefit. Because the code cannot anticipate exactly what every user will do next, it is common for such techniques to download extra data, not all of which is actually needed, to many or all clients.

3.3.5 Shortcomings

Shortcomings and restrictions associated with RIAs are:

1. **Sandboxing.** Because RIAs run within a sandbox, they have restricted access to system resources. If assumptions about access to resources are incorrect, RIAs may fail to operate correctly.
2. **Disabled scripting.** JavaScript or another scripting language is often required. If the user has disabled active scripting in their browser, the RIA may not function properly, if at all.
3. **Client processing speed.** To achieve platform independence, some RIAs use client-side scripts written in interpreted languages such as JavaScript, with a consequential loss of performance (a serious issue with mobile devices). This is not an issue with compiled client languages such as Java, where performance is comparable to that of traditional compiled languages, or with Flash movies, in which the bulk of the operations are performed by the native code of the Flash.
4. **Script download time.** Although it does not have to be installed, the additional client side intelligence (or client engine) of RIA applications needs to be delivered by the server to the client. While much of this is usually automatically cached it needs to be

transferred at least once. Depending on the size and type of delivery, script download time may be unpleasantly long. RIA developers can lessen the impact of this delay by compressing the scripts, and by staging their delivery over multiple pages of an application.

5. **Loss of integrity.** If the application-base is X/HTML, conflicts arise between the goal of an application (which naturally wants to be in control of its presentation and behavior) and the goals of X/HTML (which naturally wants to give away control). The DOM interface for X/HTML makes it possible to create RIAs, but by doing so makes it impossible to guarantee correct function. Because an RIA client can modify the RIA's basic structure and override presentation and behavior, it can cause failure of the application to work properly on the client side. Eventually, this problem could be solved by new client-side mechanisms that granted an RIA client more limited permission to modify only those resources within the scope of its application. (Standard software running natively does not have this problem because by definition a program automatically possesses all rights to all its allocated resources).
6. **Loss of visibility to search engines.** Search engines may not be able to index the text content of the application.
7. **Dependence on an Internet connection.** While the ideal network-enabled replacement for a desktop application would allow users to be "occasionally connected" wandering in and out of the hot-spots

or from office to office, today (in 2008) the typical RIA requires network connectivity.

8. **Accessibility.** There are a lot of known Web accessibility issues in RIA, most notably the fact that screen readers have a hard time detecting dynamic changes (caused by JavaScript) in HTML content.

3.3.6 Software development complications

The advent of RIA technologies has introduced considerable additional complexity into Web applications. Traditional Web applications built using only standard HTML, having a relatively simple software architecture and being constructed using a limited set of development options, are relatively easy to design and manage. For the person or organization using RIA technologies to deliver a Web application, their additional complexity makes them harder to design, test, measure, and support.

Use of RIA technology poses several new service level management (SLM) challenges, not all of which are completely solved today. SLM concerns are not always the focus of application developers, and are rarely if ever perceived by application users, but they are vital to the successful delivery of an online application. Aspects of the RIA architecture that complicate management processes are:

1. **Greater complexity makes development harder.** The ability to move code to the client gives application designers and developers far more creative freedom. But this in turn makes development

harder, increases the likelihood of defects (bugs) being introduced, and complicates software testing activities. These complications lengthen the software development process, regardless of the particular methodology or process being employed. Some of these issues may be mitigated through the use of a web application framework to standardize aspects of RIA design and development. However, increasing complexity in a software solution can complicate and lengthen the testing process, if it increases the number of use cases to be tested. Incomplete testing lowers the application's quality and its reliability during use. One could argue that the above comment applies not specifically to RIA technology, but to complexity in general. For example, that exact same argument was used when Apple and Microsoft independently announced the GUI in the 1980s and perhaps even when Ford announced the Model T. Nonetheless, humans have shown a remarkable ability to absorb technological advances for decades, if not centuries.

2. **RIA architecture breaks the Web page paradigm.** Traditional Web applications can be viewed as a series of Web pages, each of which requires a distinct download, initiated by an HTTP GET request. This model has been characterized as the Web page paradigm. RIAs invalidate this model, introducing additional asynchronous server communications to support a more responsive user interface. In RIAs, the time to complete a page download may no longer correspond to something a user perceives as important, because (for example) the client engine may be prefetching some of the downloaded content for future use. New measurement techniques

must be devised for RIAs, to permit reporting of response time quantities that reflect the user's experience. In the absence of standard tools that do this, RIA developers must instrument their application code to produce the measurement data needed for SLM.

3. **Asynchronous communication makes it harder to isolate performance problems.** Paradoxically, actions taken to enhance application responsiveness also make it harder to measure, understand, report on, and manage responsiveness. Some RIAs do not issue any further HTTP GET requests from the browser after their first page, using asynchronous requests from the client engine to initiate all subsequent downloads. The RIA client engine may be programmed to continually download new content and refresh the display, or (in applications using the Comet approach) a server-side engine can keep pushing new content to the browser over a connection that never closes. In these cases, the concept of a "page download" is no longer applicable. These complications make it harder to measure and subdivide application response times, a fundamental requirement for problem isolation and service level management. Tools designed to measure traditional Web applications may -- depending on the details of the application and the tool -- report such applications either as a single Web page per HTTP request, or as an unrelated collection of server activities. Neither description reflects what is really happening.

The client engine makes it harder to measure response time. For traditional Web applications, measurement software can reside either on

the client machine or on a machine that is close to the server, provided that it can observe the flow of network traffic at the TCP and HTTP levels. Because these protocols are synchronous and predictable, a packet sniffer can read and interpret packet-level data, and infer the user's experience of response time by tracking HTTP messages and the times of underlying TCP packets and acknowledgments. But the RIA architecture reduces the power of the packet sniffing approach, because the client engine breaks the communication between user and server into two separate cycles operating asynchronously -- a foreground (user-to-engine) cycle, and a background (engine-to-server) cycle. Both cycles are important, because neither stands alone; it is their relationship that defines application behavior. But that relationship depends only on the application design, which (in general) cannot be inferred by a measurement tool, especially one that can observe only one of the two cycles. Therefore the most complete RIA measurements can only be obtained using tools that reside on the client and observe both cycles.

3.4 TiBle

In this chapter we are presenting various details of our application called ‘TiBle’. Below we present the basic details.

3.4.1 Product Name

TiBle, an abbreviation for TIME TABLE GENERATOR is a simple graphical user interface that will allow users to easily generate the required time table for their newly enrolled batches; developed by PSSV (Priyanka, Shubham, Siddharth, Vishu) group.

3.4.2 Product Description

TiBle is a specific application for generating different types of time tables. It can give outputs for different semesters of Computer Science Department. Some of the most common constraints to deal with are listed below. Some of these are soft constraints meaning they only increase the cost.

Hard Constraints

- Range of day from 1 to 5 (Monday to Friday)
- Range of hours per day from 9.30AM to 4.50PM
- Unity of module in a timetable

- No two modules for the same course at same time
- No teacher or student must be assigned more than one class.
- There should be required number of periods for each subject (min. 5 lectures for each subject per week).
- Each course must have required consecutive periods. For example lab is assigned to 2 consecutive lectures (approx. 2 hours).
- A lunch break must be scheduled.

Soft Constraints

- Courses must be evenly distributed.
- One day free by week.
- No more than 3 hours without a break.
- Same teacher must not have consecutive periods unless specified.
- Faculty's daily lecture hours should be restricted to be within the allowed maximum hours.

3.4.3 Product Features

Some of the most important features offered by our software are:

- Simple wizard type user interface
- Login authentication with MySQL login
- Detailed Message dialogs for user assistance

- Mouse or/and keyboard for inputs
- Separate database maintaining basic information such as subjects, faculties, batches, semesters and their associations and other details
- Database for holding generated timetable and for storing required timetables.
- Features for allocating and de-allocating faculties and generated time tables

Highly efficient needs only few minutes to complete whole procedure.

CHAPTER 4

TIBLE FUNCTIONALITY

TiBle – Timetable Generator, major functionality includes, genetic algorithm for the generation of timetable. Approach used, to check for all the conflicts and then generate timetable.

4.1 Work / Data Flow

Work / Data flow of our project, first there is login interface that accepts the admin's email id and password and cross check the login table in the database for validation of entries. If entries are matched and the row select is atleast one that means there is presence of that user in table. Hence logged in. During the login process session is maintain in PHP session variable - `$_SESSION["username"]`. It has username of user. If session variable is set that means user is logged in.

Further after login it is redirected to dashboard page. Dashboard page include the name of the user from the member table and it flashes the timestamp on the dashboard. Dashboard has left menu panel that includes Generation of timetable and to view the dashboard and also to signout of account.

Clicking on Generate, redirects to timetable.php file, which accepts the year for timetable, semester, section name, number of subjects to add and number of labs to add with their corresponding faculties. After adding subjects and faculties, automation algorithm is called that focuses on the semester wise timetable. If semester is > 6 then different algorithm and if < 6 then different algorithm. Algorithm has been designed in this way that it fulfills all the constraint mentioned in our project.

4.2 Functions

Major function in our project are as follow –

- getTime() – to fetch the current time.
- Connect() and callConnect() – for connection to database.
- Login_check() – check if user is logged in or not.
- sec_session_start() – to start the session.
- unique_random() – generate unique random numbers.
- subjects() – to fetch subject list for respective semester.
- labs() – to fetch labs list for respective semester.
- Shuffle() – to shuffle the array.
- Timetable_tible() – generate timetable.
- Unallot() – unallocated faculty from subjects.
- Login() – to login into account.
- Hash() – encrypting password.
- Checkbrute – security from bruteattack
- Esc_url – security to test escape sequence attacks.
- Logout() – session termination.
- Register() – to register new account.
- changePass() – to change the password.

CHAPTER 5

Requirement Analysis

This chapter gives minimum requirement your system should have in order to make this software work. This software works in any operating system in which the developer tools or the user tools can be installed. Since we had limited resources, we could only test in Widows 7, Windows XP, Ubuntu 11.04 and Ubuntu 10.10. So usually the requirement specification will be same as that of the operating system. So we are providing a standard specification.

5.1 System Configuration

1 GHz x86 processor
1GB of system memory (RAM)
15GB of hard-drive space
Monitor to display output
Keyboard/Mouse for data input

5.2 Developer tools

Front end: PHPStorm, XAMPP with phpmyadmin
Back end: MySQL

5.3 User Tools

Web Browser

5.4 Tools for Documentation

Windows

Screenshots

Photoshop CS6

Microsoft Word

5.5 Standardized Knowledge Based Representation

Not just we wanted to implement automation process but we also wanted to keep our data in standard form, so that any external application can access our data. By doing to so we were thinking ahead of time. We were thinking about future implementation of automatic timetable generator by some other groups or by our self in other language and on other platform. Keeping knowledgebase portable, we can write cross platform application and transfer our knowledge base seamlessly between two platforms.

We decided we will go with XML. Details about choosing XML for Knowledge base are as follows.

5.5.1 XML

The Extensible Markup Language (XML) is a general-purpose specification for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet, and it is used both to encode documents and to serialize data. In the latter context, it is

comparable with other text based serialization languages such as JSON and YAML. It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML, RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages. XML is recommended by the World Wide Web Consortium. It is a fee-free open standard. The W3C recommendation specifies both the lexical grammar and the requirements for parsing.

5.5.2 Advantages of XML

1. It is text-based.
2. It supports Unicode, allowing almost any information in any written human language to be communicated.
3. It can represent common computer science data structures: records, lists and trees.
4. Its self-documenting format describes structure and field names as well as specific values.
5. The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.
6. XML is heavily used as a format for document storage and processing, both online and offline.
7. It is based on international standards.
8. It can be updated incrementally.

9. It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.
10. The hierarchical structure is suitable for most (but not all) types of documents.
11. It is platform-independent, thus relatively immune to changes in technology.
12. Forward and backward compatibility are relatively easy to maintain despite changes in DTD or Schema.
13. Its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.
14. An element fragment of a well-formed XML document is also a well-formed XML document.

CHAPTER 6

OVERVIEW

This chapter gives an overview of the system in the use case diagram, overview of the activities in the work break down diagram, overview of the working of activities in data flow diagram, and overview of the database in ER diagram.

6.1 Use Case Diagram

As shown in the figure the Database Entry Operator (DEO) is responsible of the system. He should be aware of various faculties and subjects available in the college; he should also know the various rules and regulations of the institution. His job is as follows:

- Collect teacher's information.
- Collect subject's information.
- Collect semester's information.
- Input these information to the system.
- Provide the output to others
- Make necessary modifications
- Maintain the database for future works

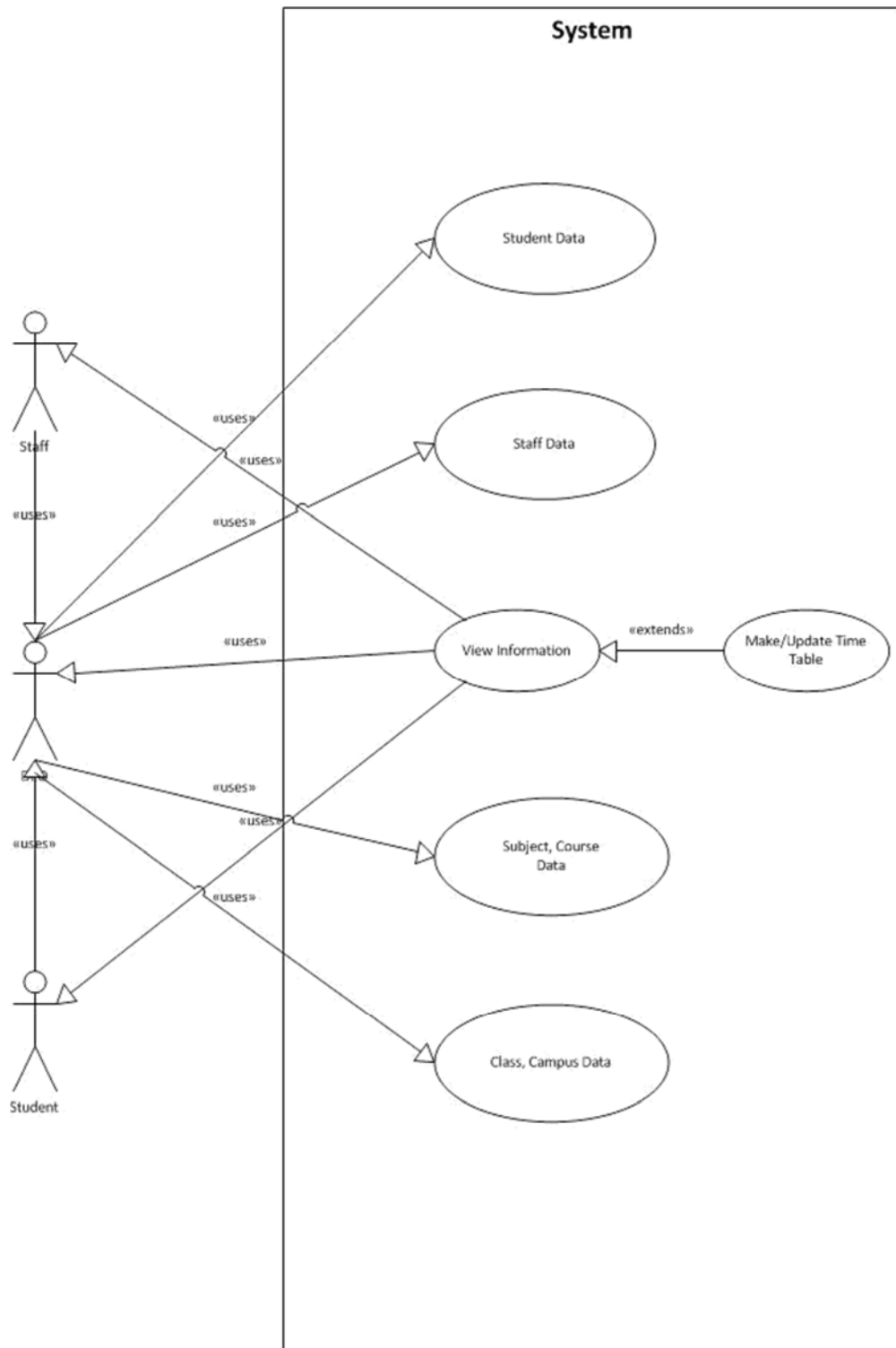


Figure 6.1 Use Case Diagram

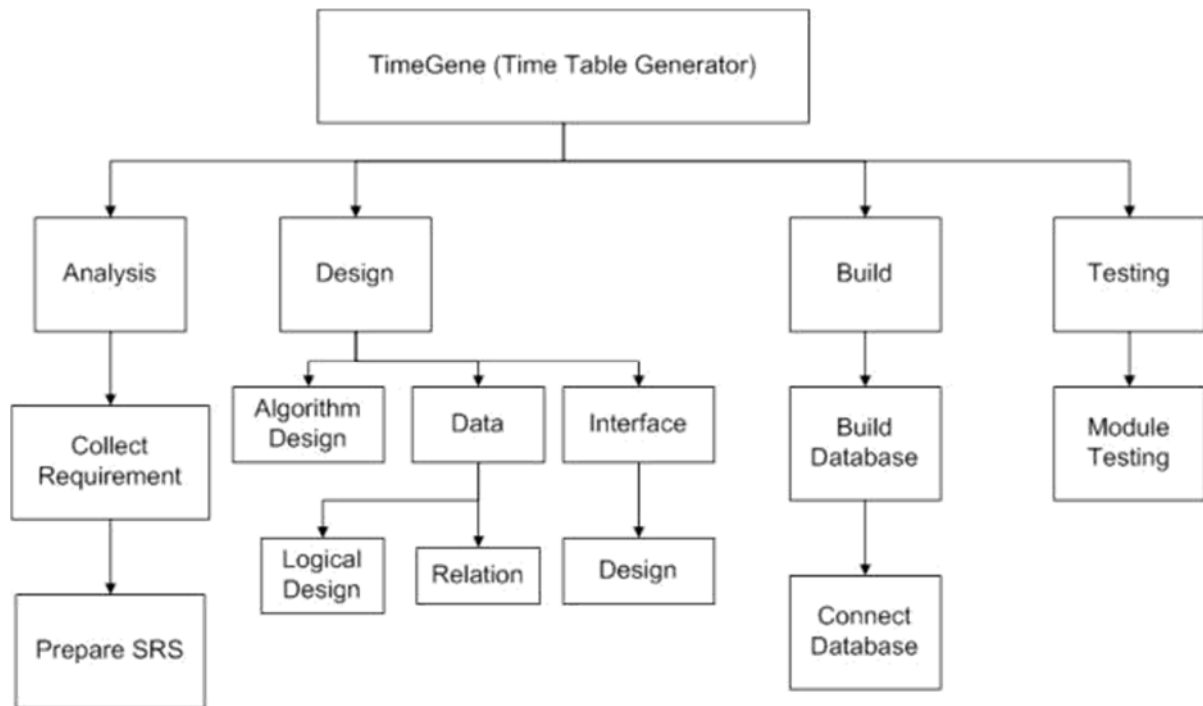


Figure 6.2 WBS(Work Breakdown Structure)

6.2 Work Breakdown Structure

Above is an activity based work breakdown structure. The diagram shows how we have divided the whole process into simpler processes. Each of the process is carried out by various group members. Most of the steps are repeated again and again in the life cycle of the project.

6.3 Data Flow Diagram

The data flow through database is described in following diagram. Each process represents the working of user with the interface and followed by

its associated operation performed in the database. Here open square represents the operation that is performed in databases.

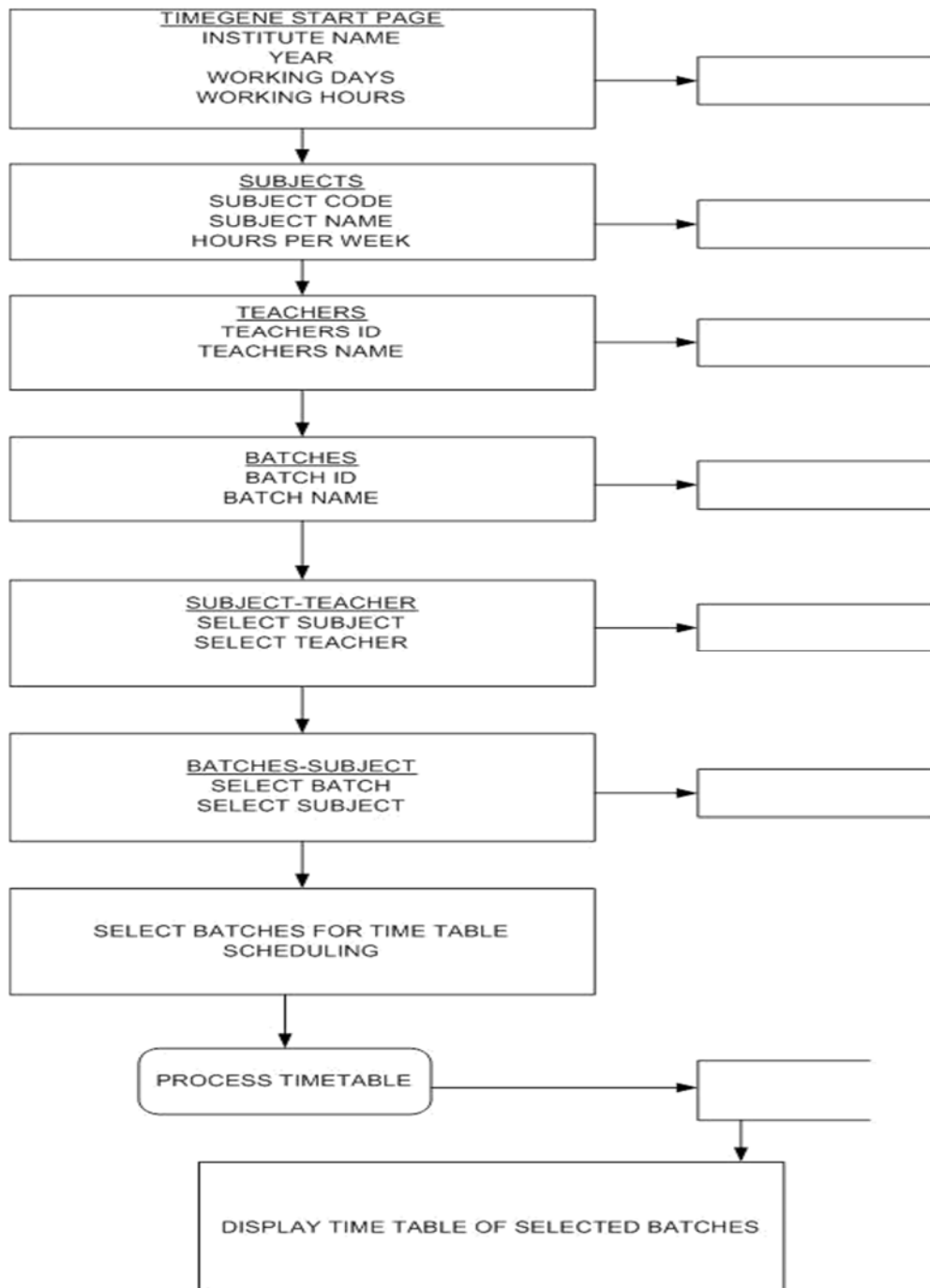


Figure 6.3 Data Flow Diagram

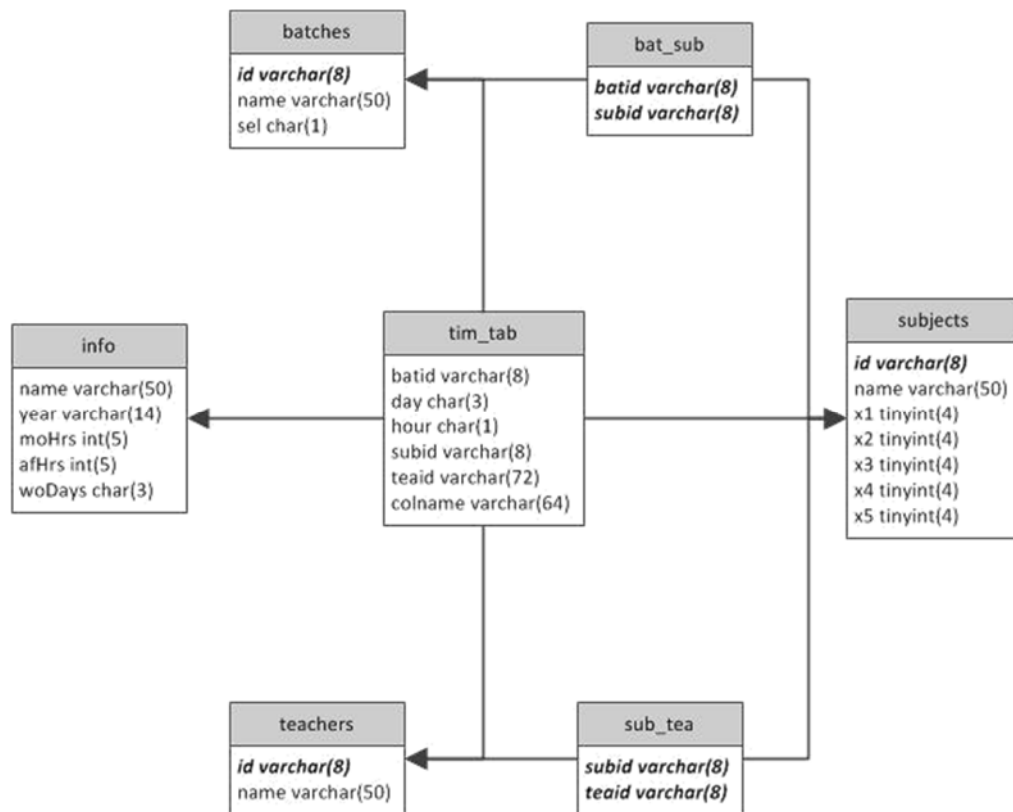


Figure 6.4 ER Diagram

6.4 ER Diagram

The database schema is represented by the ER Diagram. Here each box represents a name in the table. The table name is given at the top and the fields are given at the bottom. The bold italics fields are the primary keys used, along with field name- its data type and length are given.

They are:

- Info table stores general information such as college name, academic year, working hours and days and stores information in its associated fields in database.
- Batches table stores information of the batches such as batch id and name.
- teachers table stores information of teachers such as teacher id and name.
- subjects table stores information of subjects such as subject id, name, working hours and the number of teachers needed.
- sub tea table stores association of subjects and teachers. Here the subid is the foreign key associated with the id of the teachers table and the batid refers the id of the batches table.
- bat sub table stores association of batches and subjects. Here the batid is the foreign key associated with the id of the batches table and the subid refers the id of the subjects table.
- time table stores the generated time table and it is associated with all other tables.

CHAPTER 7

IMPLEMENTATION

This application has been developed using PHP as front end tool and MySQL Server as its back end tool.

PhpStorm has been chosen as its development environment because of the following features:

- rich and intelligent code editor for PHP
- actually "gets" your code and deeply understands its structure
- supports PHP 5.3, 5.4, 5.5 & 5.6 for modern and legacy projects
- smart code completion
- syntax highlighting
- extended code formatting configuration
- wide help and support on the web
- on-the-fly error checking
- code folding
- supports language mixtures

While coding TiBle application several constraints related to its computation has been taken into account.

Timetable generating problem provides us with various alternatives in the design of the algorithm, interface and the database. Among the various designs what we have implemented is detailed below:

7.1 Interface Implementation

There are ten classes each contains a JFrame which is associated with an interface. The associations are as follows:

- Main class for login interface
- TiBle class for basic information interface
- Subjects class for subject interface
- Teachers class for teachers interface
- Batches class for batches interface
- SubTea class for Subject Teacher interface
- BatTea class for Batch Teacher interface
- SelBat class for Batches Selection and Priority interface
- ShowTable class for Timetable Output interface
- SelectTable class for open and save interface

7.1.1 Login Interface

Enter the MySQL login name and password as primary information. Administrator, Faculties and Students have different access rights.

7.1.2 Basic Information Interface

Enter basic information related to semester, section using a drop down list. To generate new time table, click on Generate Timetable. To de-allocate generated timetable along with the allotted faculties, click on Deallocate Timetable.

7.1.3 Subject Interface

Enter the Subject ID and Subject Name, enter the number of teachers needed in subject and click Save. To edit, select the Subject ID and make the updates and click Save. To remove, select the Subject ID and click Remove.

7.1.4 Faculty Interface

Enter the Faculty ID and Faculty Name and click Save. To edit, select the Faculty ID and make the updates and click Save. To remove, select the Faculty ID and click Remove.

7.1.5 Subject – Teacher Interface

Select the Subject ID and their respective Teacher ID. Click the double headed symbol to add or remove from Selected Teachers list.

7.1.6 Batch – Subject Interface

Select the Batch ID and their respective Subject ID. Click the double

headed symbol to add or remove from the Selected Subjects list.

7.1.7 Batch Selection Interface

Select the Batches to be scheduled. Click the double headed symbol to add or remove from the Selected Batches list and if necessary set or remove subjects priority for the selected batch. The priority is set by selecting subject, day, hour from the drop down list.

7.1.8 Timetable Output Interface

Select Batch tab to view associated batch time table. Select teachers view or subjects view to display time table in teacher or subject respectively. To edit generated list select the period and edit using keyboard. To save generated time table click Save.

7.1.9 Save Table Interface

Enter filename to be saved to the database and select save. To replace an existing le or delete a le, use delete and then save the new timetable to the database. To avoid changes click cancel.

7.1.10 Open Table Interface

Select the filename and click the Open Table option. To delete an existing file, select the file and click delete. To go back press cancel.

7.2 Algorithm Implementation

Timetable generation is an NP-Complete problem; specially speaking NP-Hard. So it lacks a proper time bound for execution i.e. problems like these often can have many different outputs. So we assign cost to each output which gives the measure of deviation of the output from the desired one. So our aim is to get the output with minimum cost if there is one. Genetic algorithm can give best results but the time needed for it to compute cannot be determined so we have developed an alternative approach which can be applied to solve most of the NP-Complete problems.

- First determine the various constraints which the output must satisfy.
- We then categorize them as soft and hard constraints.
- Third step is to make a procedure which can generate an output for most of the possible inputs.
- The final step is to reduce the cost.

7.3 Database Implementation

In order to incorporate portability we use MySQL as our backend. It provides many features such as different engines and high end sql commands to create and manipulate database. It also provides tools called MySQL dump to backup database.

In our project as mentioned in the ER diagram, we use a single database called Tible which contains the above mentioned tables. Engine for table other than info and tim tab are made InnoDB for foreign key annotation, whereas info and tim tab are made MyISAM for easy editing and backup.

Primary keys are properly assigned for each table except the info as to avoid duplicate entries which may later create problems. Even though we designed GUI so as to avoid such conditions, still we don't recommend to accessing the database directly. Our algorithm assumes that the database entries are correct so any changes may cause problem.

The software when installed first checks if such a database is available, if not found it automatically creates the necessary tables. The default engine for window is InnoDB and that for linux is MyISAM these are properly addressed in our database. So the user need not worry anything about database creation.

We added a feature to save time table from generated table (tim tab). It simply copies the data in tim tab to a table with starting name saved. Besides this, a user can backup other tables or the whole database using mysql-dump command but its procedure differs on different operating systems. Later this backup can be restore.

CHAPTER 8

TESTING

Testing is an important phase in software lifecycle. Testing improves reliability and robustness of the application. The basic operations to be tested are:-

- There should be at least one working day and one working hour
- Different inputs must be checked for its range. For example no of hours in morning or evening should be between 0-5, total number of periods for a course must be between 0-70, code for course must be 0-8 character long.
- User should not be given permission to edit or modify subjects, teachers or batches without releasing its associations
- Opening and saving of databases should show exceptions if they occur (same file should not be used while saving)
- Before generating table it should be checked if all subjects are assigned at least one teacher, no of periods available ((morning hours + evening hours)*no of days) must be equal to no of periods assigned.
- If any problem occurs during generation (due to constraints) it must be properly displayed.

Test Approach

The system is to be tested at various stages of the project development:

Each user interface is tested individually for its function. Interfaces meant for data input are tested by entering data in the data tables through each interface.

Similarly each data base operation is tested through interfaces.

All the user interfaces are joined in the desired sequence and their back end coding is tested for the desired result. Like previous window close as the next desired window opens and each button performs its desired task etc.

Every class in the php code is tested individually with the help of test cases. Whole of the algorithm is tested with sample run of data to generate an optimal time table for the provided database.

Test Planning

Most of the testing requires checking connectivity of the user interfaces with the database, so a properly designed database is required for testing.

Design interfaces and connect each of them to the database and test them for proper output as is it is in the database.

Inter connect all the user interfaces in the desired sequence. Check if each of the buttons result in the desired result.

Develop the java classes for data retrieval. Test each of them according to test cases.

Develop java code for timetable generation. Test the coding with a small database by generating a time table.

8.1 Functional Test Criteria

The objective of these tests is to ensure that each element of the application meets the functional requirement of the user.

- Requirements Catalogue
- Other functional documents produced during the course of the project i.e. resolution to issues/change requests/feedback.
- Validation Testing - which is intensive testing of the new Front end fields and screens. Windows GUI Standards; valid, invalid and limit data input; screen look and appearance, and overall consistency with the rest of the application.
- Functional testing - these are low-level tests which aim to test the individual processes and data rows

8.2 Integration Testing

This test proves that all areas of the system interface with each other correctly and that there are no gaps in the data flow. Final Integration Test proves that system works as integrated unit when all the fixes are complete.

8.3 User Acceptance Test

This test, which is planned and executed by the User Representative(s), ensures that the system operates in the manner expected, and any supporting materials such as procedures, forms etc. are accurate and suitable for the purpose intended. It is high level testing, ensuring that there are no gaps in functionality.

8.4 System Test Criteria

Entrance Criteria

- All developed code must be unit tested. Unit and Link Testing must be completed and signed off by development team.
- All human resources must be assigned and in place.
- All test hardware and environments must be in place, and free for System test use.

Exit Criteria

- All High Priority errors from System Test must be fixed and tested.

8.5 Test cases and Test results

Outlined below are the main test types that are performed for this release. All test entries on wrong input has been tested to verify code stability and correctness. The test cases presented here are based on criteria presented above to validate its test implementation. Each test case table list the detailed test case results for each interface and its user inputs that can be assigned by the user to check the correctness of its implementation.

CHAPTER 9

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

TiBle application will simplify the process of time table generation smoothly which may otherwise needed to done using spread sheet manually possibly leading to constraints problem that are difficult to determine when time table is generated manually.

7.2 Future works

TiBle future works may include:

1. Student Room Scheduling

This will allow room scheduling for students in particular batches where there are multiple sections with strong strength.

2. Exam Time Table Generation

This will allow teachers/users to develop time table smoothly when multiple batches is required to hold exams.

3. Exam Room Scheduling

This will allow users to schedule rooms for students effectively.

4. Time Constraint Problem

Some institutes have large campus where travelling time is needed to be considered.

5. Generating different Choices of Time Table

By exchanging weeks or periods using Algorithm of generated timetable various choice for time table can be provided.

CHAPTER 10

SCREENSHOTS

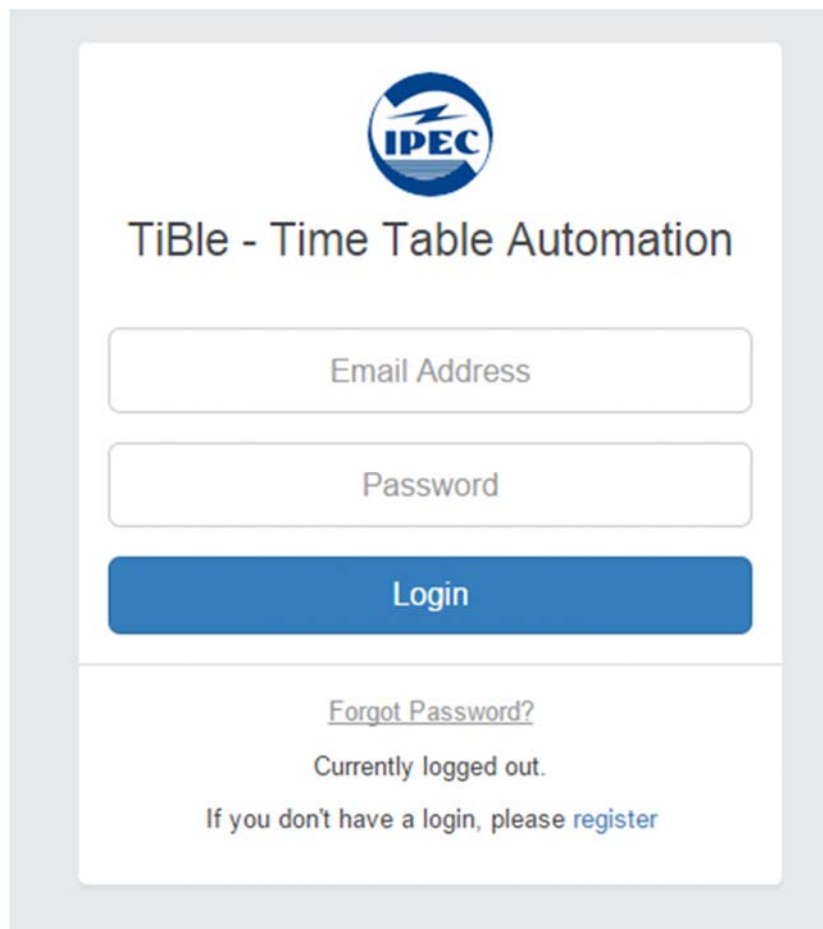

The screenshot shows a login interface for 'TiBle - Time Table Automation'. At the top is the IPEC logo, which consists of a blue circle with a white lightning bolt and the letters 'IPEC' in blue. Below the logo is the title 'TiBle - Time Table Automation'. There are three input fields: 'Email Address', 'Password', and a blue 'Login' button. Below the login button is a link for 'Forgot Password?'. At the bottom, it says 'Currently logged out.' and 'If you don't have a login, please [register](#)'.

Figure 10.1 Login Interface



TiBle - Time Table Automation

tible@ipec.org.in

.....

Login

Error Logging In!

[Forgot Password?](#)

Currently logged out.

If you don't have a login, please [register](#)

Figure 10.2 Error in Login

Sign up now

Username

Email Address

Password

Confirm Password

Sign Up

Passwords must be at least 6 characters long.
Passwords must contain -
At least one uppercase letter (A..Z),
At least one lower case letter (a..z) and
At least one number (0..9)
By signing up, you agree to the [Terms of Service](#)
Already have an account? [Sign in here](#)

Figure 10.3 Register

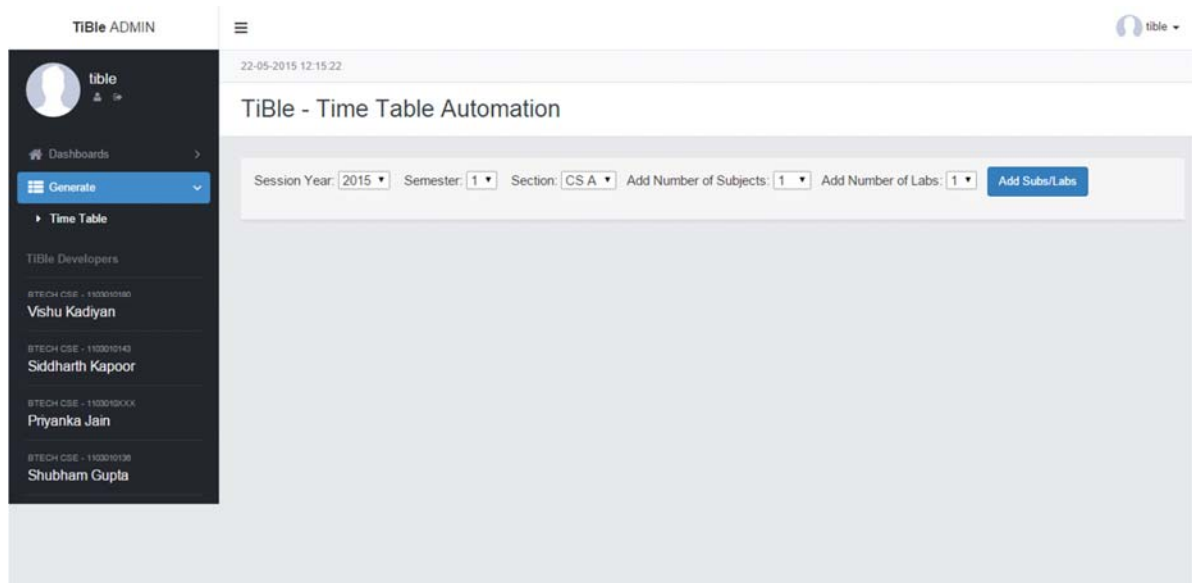


Figure 10.4 Dashboard View

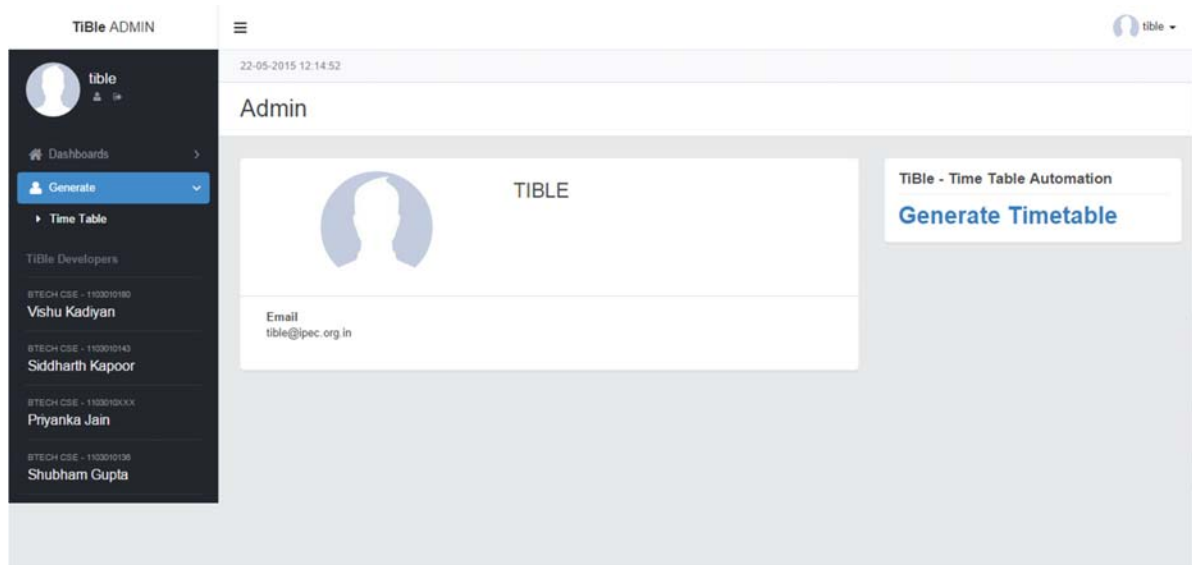


Figure 10.5 Home Page

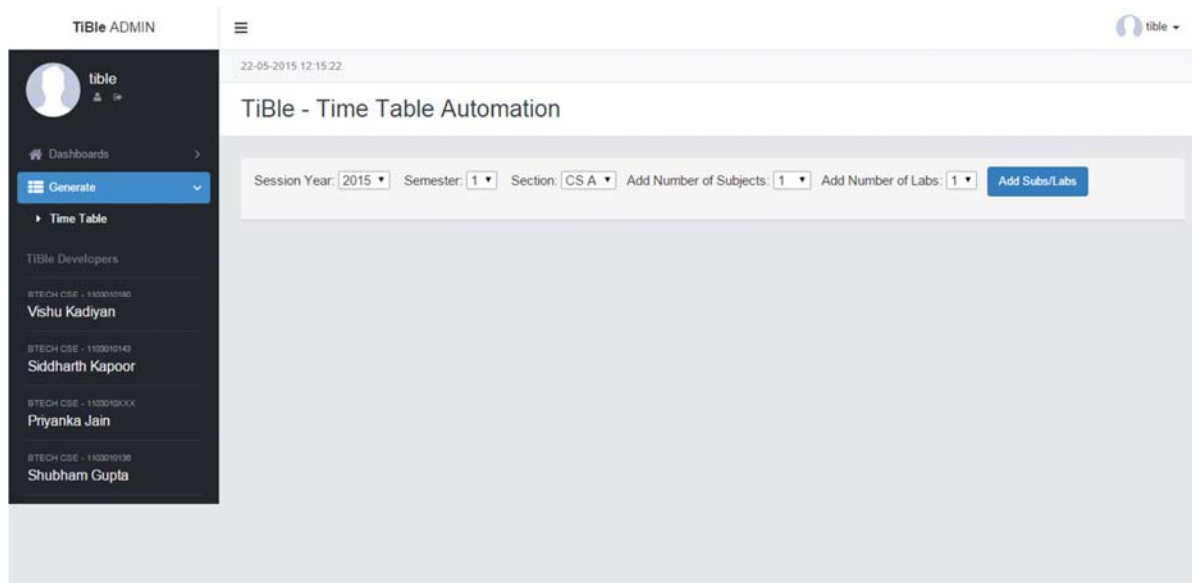


Figure 10.6 TiBle Dashboard

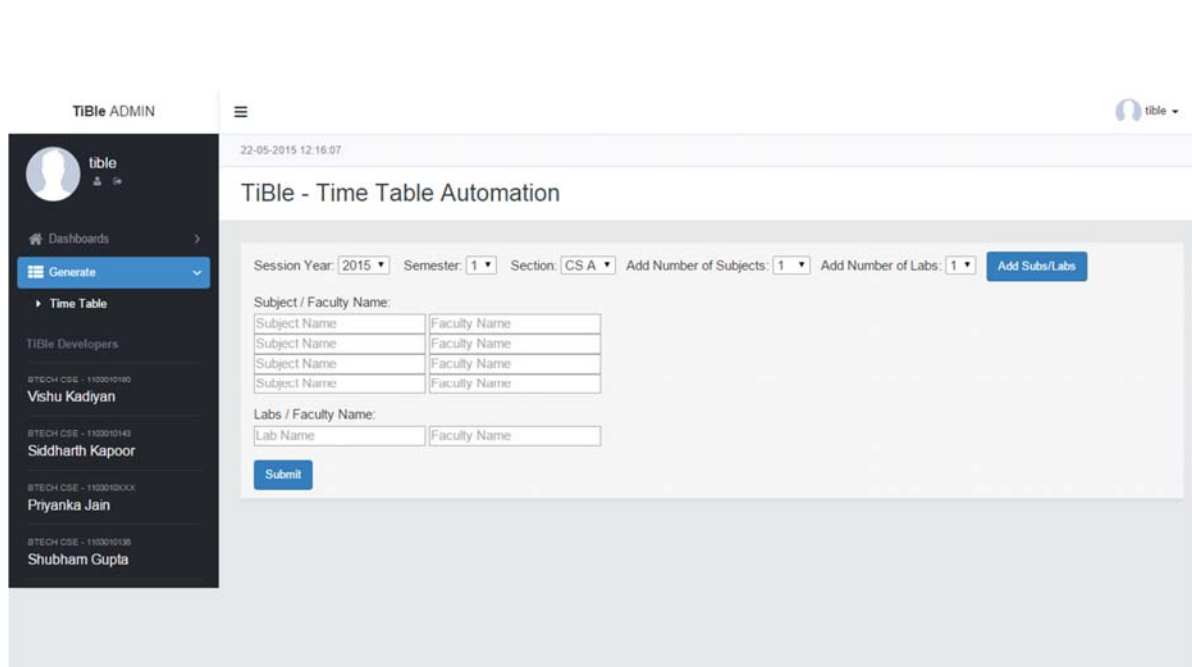


Figure 10.7 Add Subject And Labs

TiBle ADMIN

tible

22-05-2015 12:16:07

TiBle - Time Table Automation

Dashboards

Generate

Time Table

TiBle Developers

BTECH CSE - 1103010180

Vishu Kadiyan

BTECH CSE - 1103010140

Siddharth Kapoor

BTECH CSE - 1103010200X

Priyanka Jain

BTECH CSE - 1103010130

Shubham Gupta

Session Year: 2015 Semester: 1 Section: CS A Add Number of Subjects: 1 Add Number of Labs: 1 Add Subs/Labs

Subject / Faculty Name:

AI	Pooja Tripathi
NCER	Amit Yadav
SPM	Anjali
MC	Prachi Vats

Labs / Faculty Name:

AI	Pooja Tripathi
----	----------------

Submit

Figure 10.8 Filled Data For Subject And Labs

Session Year: 2015 Semester: 8 Section: CS A Add Number of Subjects: 4 Add Number of Labs: 1 Add Subs/Labs

Subjects and Faculty Name Added.

Labs and Faculty Name Added.

Generate

Figure 10.9 View After Data Added

57

Session Year: 2015 ▾
 Semester: 8 ▾
 Section: CS A ▾
 Add Number of Subjects: 4 ▾
 Add Number of Labs: 1 ▾
Add Subs/Labs

Day/Time	9.30-10.20	10.20-11.10	11.10-12.00	12.00-12.50	1.30-2.20	2.20-3.10	3.10-4.00	4.00-4.50
Monday	SPM	SPM	NCER	SPM				
Tuesday	SPM	MC	AI	SPM				
Wednesday	MC	MC	AI	NCER				
Thursday	SPM	AI	AI	SPM				
Friday	SPM	NCER	SPM	SPM				

Figure 10.10 8th Semester Time Table Generated

T

☰

tile

▼

22-05-2015 12:25:09

☰

TiBle - Time Table Automation

Session Year: 2015 ▼ Semester: 6 ▼ Section: CS C ▼ Add Number of Subjects: 6 ▼ Add Number of Labs: 3 ▼ [Add Subs/Labs](#)

Day/Time	9.30-10.20	10.20-11.10	11.10-12.00	12.00-12.50	1.30-2.20	2.20-3.10	3.10-4.00	4.00-4.50
Monday	CD	IM	CN	CD	CD	CN	ISCL	SE
Tuesday	ISCL	IM	ISCL	ISCL	SE	WT	WT Lab	WT Lab
Wednesday	SE	CD	IM	SE	CD Lab	CD Lab	SE	CN
Thursday	ISCL	SE	WT	WT	SE	IM	ISCL	ISCL
Friday	CN	CN	CN Lab	CN Lab	IM	IM	WT	WT

Figure 10.11 Broad View

TiBLE - Time Table Automation								
Session Year: 2015 ▾ Semester: 6 ▾ Section: CS C ▾ Add Number of Subjects: 6 ▾ Add Number of Labs: 3 ▾ Add Subs/Labs								
Day/Time	9.30-10.20	10.20-11.10	11.10-12.00	12.00-12.50	1.30-2.20	2.20-3.10	3.10-4.00	4.00-4.50
Monday	CD	IM	CN	CD	CD	CN	ISCL	SE
Tuesday	ISCL	IM	ISCL	ISCL	SE	WT	WT Lab	WT Lab
Wednesday	SE	CD	IM	SE	CD Lab	CD Lab	SE	CN
Thursday	ISCL	SE	WT	WT	SE	IM	ISCL	ISCL
Friday	CN	CN	CN Lab	CN Lab	IM	IM	WT	WT

Figure 10.12 6th Semester Time Table

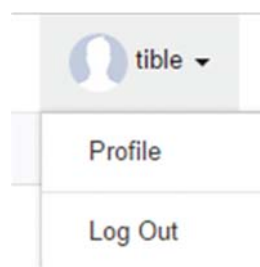


Figure 10.13 Top Right Drop Down Menu

CHAPTER 11

APPENDIX

11.1 Installation Manual

- Install XAMPP.
- Configure Apache Server for ports.
- Open localhost/phpmyadmin to configure MySQL
- Copy folder that contains code in folder C:\xampp\htdocs\ and name the folder as Tible.
- Upload .sql file in localhost/phpmyadmin import section to create database named – tible.
- Install browser – Mozilla Firefox, Chrome, Safari, Opera or Internet Explorer.
- Type path of folder, localhost/tible
- Enter necessary information (described in user interface scenarios). Get the desired timetable for specific semester as output.

11.2 User Interface Scenario

User Interface Scenario provide a brief preview of user interaction with software. User interface are defined in Screenshots.

CHAPTER 12

REFERENCES

Internet –

1. www.slideshare.net/imdzeeshan/time-table-mgt-system
2. www.trb.org/studies/idea/finalreports/transit/Transit39_Final_Report.pdf
3. www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

Books –

1. Automated Class Scheduling Generation in the Context of a University Timetable Information System, Kuldeep Singh Sandhu.
2. A Genetic Algorithm Based University Timetabling System, Edmund Burke, David Elliman and Rupert Weare.
3. An Evolutionary Algorithm for solving School Time-tabling problem, Calogero Di Stefano and Andrea G. B. Tettamanzi.
4. Constraint Based Timetabling, A.M. Abbas, E.P.K Tsang.