The Implementation and Enhancement of Timesheet Management
Process Based on Stored Procedure via .NET

Master Thesis

by

Anith Shaji
The Implementation and Enhancement of Timesheet Management Process Based on Stored Procedure via .NET

A Thesis Submitted in
Partial Fulfillment of the
Requirements of the Degree of

Master of Science
in
Information Engineering and Computer Sciences

by
Anith Shaji
Zechenstraße 40
47198 Duisburg
Matriculation Number:
22914

Submission Date:
28 February 2019
Abstract

Stored Procedures perform a significant role in the implementation of complex business logic in the database management system. Web-based application systems that handle a large amount of data with complex calculation and transformation rules, they prove to improve the system efficiency, maintainability, data integrity and reliability to a large extent.

This research will attempt to enhance a FoxPro based timesheet management system by implementing a web-driven Relational Database Management System with SQL based stored procedure and user-defined functions for faster and efficient data processing. The thesis explores the issues and the processing stages of the current timesheet management process and develops user-friendly responsive intranet online web-application portal with minimal processing steps. That accomplished by writing the processing logic using stored procedures in association with the user-defined table-valued and scalar-valued function in SQL Server and integrating them with Entity Framework three-tier web application architecture in ASP.NET MVC. The final timesheet booking data generated by the stored procedure is modified to a specific format using SpreadsheetLight, an open source spreadsheet library for the .NET framework for uploading the output booking file on SAP Business ByDesign web portal. Outcomes show that using stored procedures in a comprehensive business application can enhance the application performance by lessening the processing time and processing stages.

Keywords: Stored Procedures, SpreadsheetLight, Relational Database Management System, ASP.NET MVC, SAP Business ByDesign.
Confidentiality Clause

The following Master thesis contains confidential data and information disclosed by GITEC-IGIP GmbH. It may not be disclosed, published or made known in any other manner, including in the form of extracts, before or without the explicit permission of GITEC-IGIP GmbH. The Master thesis is made available to members of the Examination Board solely for the purpose of assessment.
## TABLE OF CONTENTS

1. Introduction ................................................................................................. 1

2. Background .................................................................................................... 3
   2.1 Entity Framework ....................................................................................... 3
   2.2 Responsive Web Design Mode and Application ........................................... 6
   2.3 Model Driven Architecture ........................................................................ 7
   2.4 Model-View-Controller (MVC) Web Design Pattern .................................. 9
   2.5 Rich Internet Applications .......................................................................... 11
      2.5.1 AJAX and AJAX Development Tools .................................................. 11
      2.5.2 Microsoft ASP.Net AJAX .................................................................. 11
   2.6 Language Integrated Query (LINQ) ............................................................ 12
      2.6.1 Basic Features of C# Programming Language ...................................... 12
      2.6.2 Key Components of LINQ Framework ............................................... 12
      2.6.3 LINQ Support for Multi-Source Heterogenous Data Integration .......... 13
   2.7 Relational Database Management System ................................................ 14
   2.8 Stored Procedure in Relational Database Management Systems ............... 16
   2.9 Object-oriented Interface to Database Stored Procedures ......................... 17
   2.10 SAP Business ByDesign .......................................................................... 18
   2.11 SpreadsheetLight and Pennyweight System ............................................. 19

3. Research Methodology, System Investigation and Analysis ............................ 19
   3.1 Research Methodology .............................................................................. 19
      3.1.1 Data Collection .................................................................................. 20
      3.1.2 System Investigation ......................................................................... 20
      3.1.3 Data Analysis ................................................................................... 27
      3.1.4 Result of Analysis and Interpretation ................................................. 27
3.2 System Analysis ........................................................................................................................................... 28

3.2.1 Problems of Existing System .................................................................................................................. 29

3.2.2 Proposal of a New System ......................................................................................................................... 29

3.2.3 Benefits of The Proposed System ............................................................................................................. 30

4 System Design and Implementation ................................................................................................................ 30

4.1 Design Phase .................................................................................................................................................. 30

4.1.1 Functional Requirements .......................................................................................................................... 31

4.1.2 Non-Functional Requirements .................................................................................................................. 31

4.1.3 Use Cases ............................................................................................................................................... 32

4.1.4 Activity Chart ......................................................................................................................................... 34

4.1.5 Three Tier Architecture Design ............................................................................................................. 35

4.1.6 Database Design ...................................................................................................................................... 39

4.2 Preparation Phase ....................................................................................................................................... 40

4.2.1 Entity Framework – Environment Setup ................................................................................................. 40

4.2.2 Entity Framework – Code First Migration ............................................................................................... 41

4.2.3 Bootstrap Template and Theme Integration ............................................................................................. 42

4.3 Implementation Phase ................................................................................................................................ 43

4.3.1 Login ..................................................................................................................................................... 43

4.3.2 Add New Employee ................................................................................................................................ 45

4.3.3 Generate SAP Booking File ..................................................................................................................... 48

4.4 Documentation ............................................................................................................................................ 58

5 Results and Discussion .................................................................................................................................. 59

5.1 Employee Work Hour Distribution ................................................................................................................ 59

5.2 Average Execution Time ............................................................................................................................... 61

5.3 SQL Server Execution Times ......................................................................................................................... 62
6 Conclusion, Evaluation and Recommendations ................................................................. 64
6.1 Conclusion .......................................................................................................................... 64
6.2 Evaluation of the System ................................................................................................... 64
6.3 Recommendation for Further Research ........................................................................... 66
References ............................................................................................................................... 67
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th></th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>2</td>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>3</td>
<td>BLL</td>
<td>Business Logic Layer</td>
</tr>
<tr>
<td>4</td>
<td>BO</td>
<td>Business Objects</td>
</tr>
<tr>
<td>5</td>
<td>DAL</td>
<td>Data Access Layer</td>
</tr>
<tr>
<td>6</td>
<td>DTO</td>
<td>Data Transfer Object</td>
</tr>
<tr>
<td>7</td>
<td>EF</td>
<td>Entity Framework</td>
</tr>
<tr>
<td>8</td>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>9</td>
<td>LINQ</td>
<td>Language Integrated Query</td>
</tr>
<tr>
<td>10</td>
<td>MDA</td>
<td>Model Driven Architecture</td>
</tr>
<tr>
<td>11</td>
<td>MVC</td>
<td>Model View Controller</td>
</tr>
<tr>
<td>12</td>
<td>OOP</td>
<td>Object Oriented Programming</td>
</tr>
<tr>
<td>13</td>
<td>ORM</td>
<td>Object-Relational Mapping</td>
</tr>
<tr>
<td>14</td>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>15</td>
<td>SAP</td>
<td>Systems, Applications and Products in data processing</td>
</tr>
<tr>
<td>16</td>
<td>SP</td>
<td>Stored Procedure</td>
</tr>
<tr>
<td>17</td>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1: Project outline and work outline ................................................................. 2
Figure 2: Basic Application Layering with Entity Framework and Database ....................... 5
Figure 3: Model Driven Architecture (MDA) .................................................................. 8
Figure 4: Model View Controller (MVC) process (Forte, 2016, p.12).............................. 10
Figure 5: Multi-source heterogenous data integration system architecture based on LINQ ...... 14
Figure 6: Testing a Relational Database ......................................................................... 15
Figure 7: Process of Traditional Timesheet Management System based on VFP ............... 21
Figure 8(a): Sheet 1 of Zeiterfassung Document............................................................... 22
Figure 8(b): Sheet 2 of Zeiterfassung Document............................................................... 22
Figure 9(a): Loading timesheet text file in VFP user interface ........................................ 23
Figure 9(b): Conversion of timesheet text file ................................................................. 24
Figure 10: Intermediate output timesheet XLS file ........................................................... 25
Figure 11: Final SAP Output Result File ......................................................................... 26
Figure 12: Use Case Diagram ......................................................................................... 33
Figure 13: Activity Chart for Generating SAP Booking File ............................................. 35
Figure 14: Three-tier architecture .................................................................................. 36
Figure 15: SQL Server Entity Relationship Diagram ....................................................... 40
Figure 16: Login Screen ................................................................................................. 44
Figure 17: Add New Employee View .............................................................................. 46
Figure 18: SAP Booking Process .................................................................................... 48
Figure 19: Upload Zeiterfassung View Tab ..................................................................... 50
Figure 20: Generate SAP File View Tab .......................................................................... 50
Figure 21: Employee Work Hour Distribution .................................................................. 61
Figure 22: Average Execution Time of Stored Procedure ............................................... 62
Figure 23: SQL Server Execution Time of Stored Procedure ......................................... 63
LIST OF TABLES

Table 1: Example of tables a) and stored procedure b) .......................................................... 18
Table 2: Basic Application Functionalities ............................................................................. 31
Table 3: Use case diagram elements and description ............................................................. 32
Table 4: HTML Helper Methods and HTML Controls .......................................................... 37
Table 5: Use Case 1 - Login ..................................................................................................... 43
Table 6: Use case 2 – Add New Employee ............................................................................. 45
Table 7: Use case 3 – Generate SAP Booking File ................................................................. 49
Table 8: Input File Data ......................................................................................................... 59
Table 9: Output Booking Data ............................................................................................... 60
1 Introduction

The modern era of the Internet has opened the means for web applications for instant information broadcasting and various other transactions, including those having financial repercussions. Currently, a large number of web applications are developed using two major technologies, Object Oriented Programming (OOP) and Relational Database Management Systems (RDBMSs). These two technologies even being uneven, best suits each other (Armas et al. 2017, p.500).

The mainstream web application that interface with databases needs to work with both object-oriented and relational data models (Mehra et al. 2007, p.283). Java and C# are the two major object-oriented programming languages for writing the business logic of an application whereas relational databases like SQL Server and MySQL control the continuous storage of data. Object-relational mapping (ORM) is a programming technique that eliminates the differences that exist between the object-oriented and relational world. It uses the object-oriented programming languages to transform data between incompatible type systems, hence allowing to address, access and manipulate objects without considering how those objects relate to their data source. In the relational database, Stored Procedures (SP) play a significant part. They enhance the flexibility of system design, support in data layer simplification and improve the efficiency in the implementation of the application (Liu 2010, Introduction, para. 1).

Stored procedures are blocks of code in the database server, which is precompiled\(^1\); it provides faster execution and reduces the network traffic (C-sharpcorner.com, 2018). This work will consist of using the SQL Server stored procedure in Code First Approach of Entity Framework in MVC 5 web application, for faster data transformation and modelling.

Developer-friendly opensource library Spreadsheetlight is compatible with Microsoft Excel and LibreOffice Calc along with SAP template. These are used at the server side to generate the final timesheet excel report. This report is uploaded using the Microsoft Excel interface and verified in SAP Business by Design web portal.

GITEC-IGIP GmbH company provides cross-disciplinary consulting services support in development projects and constitutes a permanent network of 15 companies in Africa, Asia,

---

\(^1\) compiled initially to be used again and again.
Europe and South America (Gitec-consult.eu, 2019). The existing timesheet management system in the company is developed based on Visual FoxPro, a discontinued version of Microsoft. The system takes the input timesheet file of the employees and generates the output SAP excel data file, through a batch of steps. It is in this phase where the work accomplished in this thesis is interesting.

The proposed web application system generates the output template file instantly bypassing all the batch of steps (used by the existing timesheet management system) just after the upload of the input data file using the stored procedure and functions associated with it.

![Diagram](image)

**Figure 1: Project outline and work outline**

The work done in this thesis consists of three phases, particularly a literature study phase, a system architecture design phase and finally the implementation phase. These phases typically run in parallel to each other. In the literature study section, the first goal was to explore and become familiar with entity framework concepts for efficient storage and access of data from the database, Bootstrap framework for responsive web design and AJAX technique for creating fast
and dynamic web pages. Further investigation of SQL stored procedures and functions for faster and efficient data processing and modelling.

In the second phase, the three-tier web application architecture is set up with layering in ASP.NET MVC 5 with Entity Framework and SQL Server. The user interface design of the application is implemented using a responsive bootstrap template.

The implementation phase consists of server-side C# coding, the creation of a stored procedure with its associated set of user-defined function and a testing stage. In the testing stage, the analysis of the performance and efficiency of the system is done, by monitoring the time taken to generate the legitimate output file for the uploaded input data file of each month and further uploading them to the SAP Business ByDesign web portal.

Figure 1 depicts the work outline for the transformation process of the input excel data file to its corresponding output SAP template file in the proposed web application system.

Chapter 2 covers Background information on the theoretical part of the different techniques used throughout this work. Chapter 3 discusses existing FoxPro based system and its drawbacks. In addition to that, its subsections detail the need for its enhancement using our proposed approach.

Chapter 4 illustrates the design and implementation steps of the system with a detailed description of the scenarios required for the generation of SAP timesheet booking file. Chapter 5 presents the results of the proposed application system and discuss them. Chapter 6 closes and provides some possible future improvements to our solution.

2 Background

This section will serve as the baseline information for the central concepts used throughout this project. Firstly, we will give an overview about the concepts used in the client and the server side of the web application system, then the storage and processing mechanism within the relational database and finally about the SAP Business ByDesign and pennyweight system which will be concerned with our approach.

2.1 Entity Framework

An entity in entity framework refers to a class defined within your application. It acts as a DbSet<TEntity> type property in the derived context class. Each entity in the application system resembles a table and each property associated with these entities form the columns of the tables in the database.
For example, the following Zeiterfassung entity (domain class) is in our project and refers to the input timesheet data file.

```csharp
public class Zeiterfassung
{
    // Gets or sets the primary key
    [Key]
    public int ZeiterfassungID { get; set; }
    // Gets or sets the attributes for Zeiterfassung
    public int PersonalNummmer { get; set; }
    public string Stunden { get; set; }
    public string Zeitart { get; set; }
    public string Kostentrager { get; set; }
    public string Aktiv { get; set; }
    public string Month { get; set; }
    public string Year { get; set; }
}
```

The class Zeiterfassung has attributes, ZeiterfassungID as its primary key, PersonalNummmer of integer type and Stunden, Zeitart etc. of string type. When the above class instance (Zeiterfassung) added to the DbSet<TEntity> property of a context class, it is then referred to as an entity, as shown below:

```csharp
public partial class GITEC_TimeSheetContext : DbContext
{
    public GITEC_TimeSheetContext ()
    : base("name=GITEC_TimeSheetContext")
    {
        Database.SetInitializer(new MigrateDatabaseToLatestVersion<GITEC_TimeSheetContext, Migrations.Configuration>());
    }
    public DbSet<Zeiterfassung> Zeiterfassung { get; set; }
}
```

The context class GITEC_TimeSheetContext with Zeiterfassung as the property of type DbSet<TEntity> and referred to as an entity set. The entity Zeiterfassung is referred here as the entity type.
Entity framework is an open source Object Relational Mapper for .NET, supported by Microsoft. It automates the process of storage and access of data from the database. The framework allows decreasing the impedance discrepancy of the application and data-centric services by raising the level of abstraction from the relational level to conceptual (entity) level (Adya et al. 2007, p. 877). The official definition of Entity Framework given by Microsoft: “Microsoft ADO.NET Entity Framework is an Object/Relational Mapping (ORM) that enables .NET developers to work with the relational database with .NET objects. It eliminates the need for writing most of the data access code that usually developers need to write. In the server-side coding, developers write LINQ queries, then retrieve the data and process them as strongly typed .NET objects” (Wadhwa 2018, p.157). Zeiterfassung document contains the cumulative timesheet data of the employees and form the input file for the system for timesheet data processing and represented within the entity framework in the form of Zeiterfassung entity (domain class).

![Diagram](entityframeworktutorial.net, n.d.)

Figure.2: Basic Application Layering with Entity Framework and Database

In Figure .2, we can see how Entity Framework integrated into the data layer of the application between the business entities (domain classes) and the underlying database. Here the UI refers to the User Interface layer and acts as the presentation interface of the application system for the users.
2.2 Responsive Web Design Mode and Application

Responsive Web Design (RWD) mainly focus on the perfect design layout of the web pages to satisfy a variety of internet terminals resolutions. Wei Jiang et al. (2014: pp.1303-1304) have presented how to achieve responsive web design by using Media Query, Bootstrap responsive navigation and flow layout.

**Media Queries**

Media Query is a CSS approach introduced in CSS3; this module allows content rendering based on the screen resolution (like smartphone or laptop screen). Among different types of media properties, the responsive design considers the attributes, that can determine the size of the screen like max-width or min-width and determine the orientation of the device (Wei Jiang et al. 2014, p.1304). The example code for different output resolutions shown below:

a) Desktop and Laptops
   
   ```css
   @media only screen and (min-width: 1024px) {...}
   ```

b) iPad
   
   ```css
   @media only screen and (min-width:768px) and (max-width:1024px) {...}
   ```

c) iPhone
   
   ```css
   @media only screen and (min-width:320px) and (max-width:767px) {...}
   ```

d) Smartphones
   
   ```css
   @media only screen and (min-device-width:320px) and (max-device-width:480px) {...}
   ```

**Bootstrap Responsive Navigation**

Bootstrap is an open source package used for front-end CSS framework, that uses CSS and HTML based design templates and written in dynamic CSS language. The additional JavaScript extension for bootstrap allows creating switchable navigation for different screen resolutions. The responsive style sheet is included in the header to make bootstrap to attain the responsive navigation, by using the code below:

```
<link href="assets/css/bootstrap-responsive.css" rel="stylesheet">
```

**Flow Layout**

The web content width expressed as a percentage format in the flow layout. The two major feature in this layout include:

1) Set float: left; to all the DIV elements present within the layout.

2) Express width in percentage.
For example, we define a CSS rule: `div#body_content {width:80%;}

Here `div` having ID `#body_content` take the width of 80% of its parent element. By using the above concepts, web application developers can easily design the responsive user interface for multiple device platforms (like iPad, mobile phone desktop computers etc.).

2.3 Model Driven Architecture

Model Driven Architecture (MDA) or Model Driven Development (MDD) is an approach to capture the system logic during the development face of an end-to-end enterprise application (see Figure 3). A model refers to a diagram that represents the working of the system before the code generation. These models of software that abstract the technology-specific information are consistently altered to deployable technology-specific implementations, as presented in (Matheson et al., 2004).

Model Driven Development Phases

Multiple phases that built up the MDD methodology includes (Larrucea et al. n.d., section 2, para. 4):

1. **Capture User Requirements**: Identify the functional and non-functional requirements of the customer and to formalize these requirements into an Application Model and further to generate an initial Application PIM (Platform Independent Model) and functional requirement specification based on a common framework of reusable assets.

2. **PIM Context Definition**: The main aim of this phase is to define the system goals and business principles. In addition to that to describe the external actors or the stakeholders using the system and to specify the required business events and to exchange the business objects.

3. **PIM Requirement Specification**: The main activities involved in this phase are to refine PIM Context and to determine the events, services and business objects or entities (refer section 2.1), produced and consumed by the application and the actors interacting with the system. Specification of the use cases, non-functional requirements and atomic requirements.
4. **PIM Analysis**: In this phase, the internal view of the system is modelled without considering the technology and retaining the separation between the functional and non-functional aspects.

5. **Design**: This phase focus on determining the system implementation and architectural style, pattern and platform used. In the first stage platform independent solution for the identified requirement is designed. The PIM elements commonly incorporate the software components like the user interface, port etc. Whereas in the second stage platform specific solution is created by refining the platform independent solution. The PSM is platform specific (for instance .NET, JAVA) and fully support the coding and deployment of the application.
6. **Coding and Implementation**: The objective of this phase is to develop the components and classes (or entities), organize the code, perform unit testing and integration of the components and subsystems.

7. **Testing**: Testing activities are performed in this phase mainly, to generate test cases or test scripts, execute the tests, correct issues and finally prepare the test document.

8. **Deployment**: In this phase the transition of the developed application to the customer. It includes activities like creation of product manual, installation or hosting of the product in the client premises (for instance hosting of a web application on client-server).

2.4 **Model-View-Controller (MVC) Web Design Pattern**

F. Grove and Ozkan (2011: p.127) present an MVC pattern that more accurately describes how MVC implemented in web frameworks. They form the basis for several web application frameworks like ASP.NET, Rails and Struts.

The three major functional components of the MVC design pattern are (Forte, 2016, p.11):

- **A. Controller**: handles the information flow by accepting the user inputs from the view page and notify both model and view to carry out the actions based on that information.

- **B. Model**: manages the (CRUD) database operations mainly create, read, update and delete and maintains the data routines in the application.

- **C. View**: represents the user interface layer and presents the data contained in the model. It includes mark-up displayed as web pages.

In general, the object-oriented design patterns are of three types – creational pattern, structural pattern and behavioral pattern. MVC pattern is a front controller pattern and fall under the category of the structural design. Several web development framework use MVC like Spring for Java programming language, Django for Python programming language and ASP.NET MVC for C# programming language.
Figure 4: Model View Controller (MVC) process (Forte, 2016, p.12)
The above-mentioned functional components in an MVC-Web pattern interact in the following manner (F. Grove and Ozkan 2011, p.130):

A. Model-View: View elements query the model to retrieve the essential information needed to display in the user interface.

B. Model-Controller: Action elements of the controller invoke model elements to perform requested transactions. Controller requests data from the model and transfer to the view component.

C. Controller-View: Controller prepare the required information for the view elements. It also identifies the view element to be displayed to the user as a response for the underlying request.
2.5 Rich Internet Applications

Lawton (2008: p.10) details about the techniques used to construct web applications having enhanced features and functions similar to the stand-alone desktop software. They provide higher interactive capacity with responsive user interfaces, improves the performance of the traditional web applications and makes the internet-based programs smooth to use and more functional.

2.5.1 AJAX and AJAX Development Tools

AJAX stands for Asynchronous JavaScript and XML and is used to create interactive web applications. When a form is submitted using AJAX, the application makes a server call, fetches new data and updates the required portion of the webpage instead of reloading the entire contents, that improves the performance of the web application, responsiveness and interactivity. The XML fundamentally specifies the data format of the server data (any format including text is accepted).

In the market, AJAX based platforms are now available from several vendors and organizations. The history of the software industry clearly illustrates the importance of platforms and the threat the Internet represents to these platforms. Therefore, businesspeople are urging to introduce platforms that leverage the network by promoting their specific tools and technologies in an attempt to secure their future relevance, said Stephen O’Grady, an analyst with market-research firm RedMonk (cited after Lawton 2008, pp.10-11).

AJAX works in combination with loosely typed JavaScript, CSS for separation of content and presentation style, XMLHttpRequest JavaScript object for asynchronous server interaction. Add it as a plug-in to the IDE (Integrated Development Environment) like Microsoft Visual Studio.

2.5.2 Microsoft ASP.Net AJAX

Microsoft ASP.Net AJAX enables the .NET developers to create responsive web pages quickly using familiar user interface elements. AJAX extensions allow Visual Studio applications to run efficiently on all browser platforms with an AJAX engine and also to access an ASP.Net server. With these extensions, .NET developers can leverage their existing skills, tools, and server infrastructures to create AJAX applications, explained Chris Swenson, Microsoft’s senior.NET platform strategy manager (as cited in Lawton 2008, p.11). In an overall view, back end Rich Internet Applications (RIA’s) run by the server, the client code generated by the libraries access the application through the browser and the programs are written in the development framework.
2.6 Language Integrated Query (LINQ)

Wang and Zheng (2009: p.1464), have designed a mini-course of Language Integrated Query (LINQ) programming that minimizes the existing gap between the traditional OOP and Database Programming (DBP). LINQ extends its data capabilities to .NET languages like C# and Visual Basic by modelling different types of data through OOP principles.

2.6.1 Basic Features of C# Programming Language

The .Net language C# make use of some of the major concepts for server-side (Wang and Zheng 2009, p.1465), which mainly includes:

1) **Enumerators**: an enumerator is similar to that of a pointer object that maintains the record of the indexes of the collection. `IEnumerable` or `IEnumerable<T>` interface can also be implemented over the user-defined collection to perform `foreach` enumeration with their support.

2) **Local type inference**: `var` keyword is used to define a variable without specifying its data type and allowing dynamic type assignment to the variable during the compilation of the initialization expression.

3) **Lambda Expression**: used for constructing LINQ query expressions and declares the method code inline.

2.6.2 Key Components of LINQ Framework

The main components in the LINQ framework (Wang and Zheng 2009, pp.1466-1467) include:

1) LINQ to Object: `IEnumerable` or `IEnumerable<T>` any collection support LINQ queries, without the use of any API or intermediate provider of LINQ. Enumerable collections like `List<T>`, `Array` or `Dictionary<Key, Value>` can be queried using LINQ.

2) LINQ to XML: .Net Language Integrated Query (LINQ) framework make use of the in-memory XML programming interface by LINQ to XML component.

3) LINQ to ADO.NET: used to query enumerable objects like DataSet (LINQ to DataSet), SQL server database schema (LINQ to SQL) and entity data model (LINQ to Entity) in ADO.NET.

The back-end or server-side developers, find it easy to fetch data from different data sources using these simple LINQ query. It makes the coding faster for the application programmers and improves the data transformation between the application and its underlying data source.
2.6.3 LINQ Support for Multi-Source Heterogenous Data Integration

Wang and Zhang (2012: p.147), have presented a heterogenous multi-source data integration technology based on LINQ. The three-layer structure designed for the software system by using this technology add the decomposed query logic and the query result with the business logic layer and from the underlying heterogeneous data layer. All the data sources are accessible via the data access layer, where the LINQ provider acquire the required data of unified format for data integration.

Multi-source heterogenous Data Integration designed based on LINQ

Figure 5 extracted from (Wang and Zhang 2012, p.148), shows the four-layer architecture of the system with,

1) the heterogeneous data layer,
2) the data access layer based on LINQ,
3) the business logic layer of query processing and
4) the top layer is the user interface or presentation layer.

The system uses a three-tier design pattern which helps in the separation of the Business Logic Layer (BLL) from Data Access Layer (DAL) so that it provides a flexible structure for expansion of system functions and improves the performance.

The bottom data layer consists of all data source supported by LINQ such as data structures, Web services, file system, SQL database, Dataset in ADO.NET and .NET collections. In the Data Access Layer, the decomposed query logic accepted by the query execution module and generates an intermediate code which transferred into the LINQ provider. Finally, the generated query result module transforms the query result to a standard format and then pushes it to the caching module. In the Business Logic Layer, the query logic module generates a standardized global query logic based on the queries from the user interface. The global query block decomposed by the decomposition query logic layer along with the logical query template and the generated sub-queries sent to the query execution module. Query results generated from the generating query results module is cashed into temporary tables or objects by the caching query results module. The cached result is further restructured by integrating the query result module and gets the integrated query data based on the query submitted by the interface layer and then returns the final output to the user interface layer. In most of the web application development, developers prefer using the three-tier architecture system integrated with the concept of LINQ. It
provides greater modularity with better unit testing features for the application programmers. This layering concept is used during the initial architecture design stage of our system, having the relational database only considered in the Heterogeneous Data Layer (as shown in Figure 5) which we refer simply as Data Layer.

![Diagram](image)

Figure 5: Multi-source heterogeneous data integration system architecture based on LINQ (Wang and Zhang 2012, p.148)

2.7 Relational Database Management System

The paper by Strawn and Strawn (2016: pp.63-64), describes the relational model concept in a database system and portrays its evolution. The first practical working RDBMS, named as Ingres (Interactive graphics and retrieval system) developed by Michael Stonebraker, implemented on minicomputers. The relational model perfectly separates the data, represented in the form of tables and the database management system in the computer’s memory handles the storage of these tables. A data table consists of rows or tuples that refer to unique entities (such as the student) and columns or attributes that correspond to various properties associated with those entities (such as student id, name, department and grade). Primary key refers to a key attribute value (like student id) that uniquely identifies each record or row in a table. Relational Database
Management System (RDBMS), contains a collection of tables and these tables can be related to each other using foreign keys. A foreign key in a table refers to the primary key of another table for cross-referencing. The join operation between multiple tables, performed with the help of a foreign key. The significant operations like insert, update and delete (on a row of a table) were available in a relational model with the addition of SQL. Now the relational database is more widely used for bulk data storage purpose by several web application portals.

**Test-Driven Development**

Developers perform database testing both at internal and interface level using clear-box also referred to as white-box testing and black-box testing respectively. Ambler (2007: pp.37-38), explains the extension of Test-Driven Design (TDD) to database development.

Relational Database Management System, implements several business functionalities and also developers write code in the database in the form of stored procedures, functions, triggers etc. Hence, developers have identified the need to perform database regression testing to improve the database quality and to maintain the integrity even after any modification made in the database (like updating stored procedure code or changing an attribute type of a table).

![Figure 6: Testing a Relational Database](Ambler 2007, p.38)

Figure 6 shows the elements regarded during testing a relational database and the dashed lines represents the threats that exist within the database (clear-box testing) and a database interface
(black-box testing). The stored procedures or the functions, written within the database need to be tested similarly to that of standard application code.

2.8 Stored Procedure in Relational Database Management Systems

A Stored Procedure (SP) is a compiled set of SQL commands, saved on a database server and having characteristics like modularity, high efficiency, data security, reduced network traffic and Central Processing Unit (CPU) load (Shao et al. 2010, p.41). In the four-layer architecture set up of the application as illustrated in section 2.6.3, the stored procedure impersonates as an interface for the data access layer by accepting and acknowledging the request from the application system.

As mentioned by Tan Baohua and Zeng Ling (2010: p.594), for application systems handling the bulk amount of data having complex transaction rules a higher level of system security is required and is always concerned with factors like efficiency, maintainability, data integrity and reliability and all this can be achieved using the stored procedure.

Syntax for Stored Procedures in SQL Server Database

```
CREATE [ OR ALTER ] PROCEDURE [Schema_Name].Procedure_Name
    @ Parameter_Name data_type,
    ...
    @ Parameter_Name data_type
AS
    [ BEGIN ]
    sql_statement [:] [ ...n ]
    [ END ]
```

**Arguments:**

1. Schema_Name: Refer to the schema name. (like dbo.).
2. Procedure_Name: The name of the procedure and must be unique within the database schema. It must not be a reserved keyword.
3. @Parameter_Name: The parameter for the procedure and always start with @ symbol as its first character. A stored procedure can have zero or more parameters according to the user requirements.
4. **data_type**: All declared parameters must have an appropriate datatype.

5. { [ BEGIN] sql_statement [:] [ ... ] [ END ] } : The body of the procedure comprising of one or more Transact-SQL statements. The statements are enclosed using the optional BEGIN and END keywords.

The basic syntax of Stored Procedure in SQL Server database is given above and can be executed by an application program by calling its name and passing values to its parameters. Application programmers place the complex business logic code in stored procedures for integrity purpose since the business logic remains close to the data stored in the database (Mahmood et al. 2016, p. 519).

**SQL Server User-Defined Function**

Tang et al. (2016: Abstract), show the rising demand to integrate big data analytics using SQL. The paper presents user-defined SQL functions (mainly scalar and aggregate functions) that enables big data platform to expose their analytical potential to a vast end user, faster search processing and improving significant business value. A user-defined function is a collection of SQL statements or Common Language Runtime (CLR) block and performs a specific action or complex calculations based on the parameters returns the result as a value. (Docs.microsoft.com, 2019). The value returned can be in the form of a single value (scalar-valued function) or a table (table-valued function).

### 2.9 Object-oriented Interface to Database Stored Procedures

Pereira et al. (2005: p.99), describe a database as “a shared collection of logically related data and a description of this data, designed to meet the information needs of an organization.”. A stored procedure acts as a clean separating layer that separates the persistent logic on the database tier from the business logic on the business layer of the application.

Let us look at the following example given in (Pereira et al. 2005, p.102), to understand the methodology for implementing an object-oriented platform to RDBMS stored procedures, using Microsoft.Net technologies like C# and SQL Server (see Table 1).

Table 1 a) considers two class instance or tables, one tbUser having login attributes of a user and the other tbProfile that hold the profile details of the user. The application system requires two criteria for the selection of user based on

1) username and password and
2) profile identification.
Here, we can also see that the attribute or the column name \textit{tbProfile\_id} acts as the foreign key (refer section 2.7).

Table 1: Example of tables a) and stored procedure b) (Pereira et al. 2005, p.102)

<table>
<thead>
<tr>
<th>a) tables</th>
<th>b) stored procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{tbUser}</td>
<td>\textit{create procedure spUser}</td>
</tr>
<tr>
<td>\texttt{tbUser_username} varchar(15)</td>
<td>\texttt{( @tbUser_username varchar(12) = '',}</td>
</tr>
<tr>
<td>\texttt{tbUser_password} varchar(15)</td>
<td>\texttt{ @tbUser_password varchar(12) = '',}</td>
</tr>
<tr>
<td>\texttt{tbProfile_id} int</td>
<td>\texttt{ @tbProfile_id int )}</td>
</tr>
<tr>
<td>\textit{tbProfile}</td>
<td>as</td>
</tr>
<tr>
<td>\texttt{tbProfile_id} int,</td>
<td>if ( @tbProfile_id = -1 )</td>
</tr>
<tr>
<td>\texttt{tbProfile_ref} varchar(15)</td>
<td>\texttt{ select * from tbUser u}</td>
</tr>
<tr>
<td></td>
<td>where( u.tbUser_name = @tbUser_name ) and</td>
</tr>
<tr>
<td></td>
<td>\texttt{ ( u.tbUser_password = @tbUser_password )}</td>
</tr>
<tr>
<td></td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>\texttt{ select * from tbUser}</td>
</tr>
<tr>
<td></td>
<td>where ( u.tbProfile_Id = @tbProfile_id )</td>
</tr>
<tr>
<td></td>
<td>return;</td>
</tr>
</tbody>
</table>

The selection criteria mentioned in Table 1 b) is further implemented by a single SP since the return value remains the same for both the queries (i.e. the unique record in the table \textit{tbUser}). Application programmers, therefore, prefer some of the system logic to be encapsulated within the stored procedure to improve the system performance and also to increase the system maintainability because of its modular nature and independence from the object-oriented application system and its underlying relational database (Shao et al. 2010, p.44).

2.10 SAP Business ByDesign

SAP Business ByDesign is software as a service (SaaS)\(^2\) and designed for small and medium-sized companies for resource planning and business management. It is a cloud-based software sold by SAP SE, a German software company. The software handles major modules like customer relationship management, human resource management, project management, financial management, supply chain management etc. The time administration section which is a part of the Human Resource (HR) module, has the time type that defines two categories of time, working and non-working time, entered by the company employees in their respective time sheets. This help to track time worked by the employees of the company on projects and accurately remunerate the employees based on the actual working hours (Inecom, 2018).

\(^{2}\) SaaS is a software licencing and delivery model and its licence is based on subscription and is hosted centrally.
2.11 SpreadsheetLight and Pennyweight System

**SpreadsheetLight**

SpreadsheetLight is an open source developer friendly open XML spreadsheet library for .NET framework written in C# and released under the MIT License. They are suitable for creating new open XML spreadsheets and also operate with existing spreadsheet documents that are compatible with Microsoft Excel 2007/2010/2013 and LibreOffice Calc. It utilizes less amount of memory space and thus enabling the server to run other programs and tasks. The spreadsheet library supports basic worksheet manipulation like cell manipulation, rich text support, conditional formatting with formulas and bulk upload of data using data tables (Spreadsheetlight.com, 2018).

**Pennyweight System**

A pennyweight is a unit of mass and represented by the unit dwt (denarius weight). One pennyweight unit is equivalent to 24 grains, 1/20 of a troy ounce, 1/240 of a troy pound and 1.55517384 grams (Archive.org, 2018). The common usage of pennyweight is in the valuation and measurement of precious metals. Jewelers determine the price of certain precious metals used for the casting process of jewelry. The abbreviations used for penny weight are dwt, pwt and PW.

3 Research Methodology, System Investigation and Analysis

In this chapter first, we will see the research methodology adopted for data collection and review the studies carried out on the existing Visual FoxPro based timesheet management system used by the company, later we explore the problems associated with the current system and finally discuss the need for its enhancement using our proposed web application system.

3.1 Research Methodology

Research is the process of arriving at a reliable solution to a given problem through the systematic collection, analysis and interpretation of data. In this thesis, the initial research is carried out on the existing Visual FoxPro timesheet management system focusing on the type of data used within the system for processing.
3.1.1 Data Collection

There are different data collection methods present and based on the nature of the research the approach of data collection varies. The two types of data categories used in this research include primary and secondary data.

Primary Data

Primary data refers to the real-time data gathered by the researcher, relevant to the subject under investigation. The primary data source provides firsthand knowledge and the tools used in case of this project for collecting this data are interview, observation and questionnaires.

Secondary Data

Secondary data is a previously gathered research data like information that is available in printed form. Secondary data sources we employed are internal records and website (mainly SAP Business ByDesign web portal).

The Interview Approach

In the interview approach a combined format of the oral interview, questionnaires and observation method was carried out on the staffs in Finance, Information Technology and Human Resource department of the company GITEC-IGIP GmbH. The principal focus of this approach was to collect feedback about the traditional timesheet management system and also to obtain the former records of data.

3.1.2 System Investigation

The main aim of system investigation is to recognize the operation of the current system to arrive at vital and relevant facts that will serve in the design and implementation stages of our recommended system. In this section, the workflow of the existing Visual FoxPro timesheet management system is explained based on the document named “Hour Handover” (cited in Appendix 1) of the company GITEC-IGIP GmbH.

Timesheet Management System based on Visual FoxPro

Visual FoxPro (VFP) is a Microsoft data-centric object-oriented and procedural programming language with a built-in relational database (Xie and Chang 2011, Introduction, para. 1). In the year 2007, Microsoft ended the development and marketing of VFP and offered no longer technical support for VFP 9.0 in January 2010 (SearchSQLServer, 2018). The traditional system used by the company is based on VFP and follow a sequence of steps to generate the final output
SAP timesheet file. The following section describes the five main steps involved in working with the system (see Figure 7).

The finance department of the company collects the detail working hours from each employee in a document named “Stundennachweis”. The input XLSX timesheet file (also referred to as Zeiterfassung) is then created. The spreadsheet has two main worksheets, one having the time recording data and the other comprising the project details of currently ongoing projects of the company.

![Figure 7: Process of Traditional Timesheet Management System based on VFP](image)

**Step 1: Creating a Text File for Input Timesheet File**

In this step, the user receives the Zeiterfassung document from the finance department of the company and recognise the six main columns in the worksheet that include Personalnummer, Zeitart, Projektnummer, Kostentrager, Aktiv and Stunden(see Figure 8(a)). The Personalnummer is a unique number assigned to each employee within the company, Zeitart is a constant having a default string value of “Projektarbeit”. Projektnummer refers to a unique number for each ongoing project of the company and projects are validated against ProjectsTable worksheet (see Figure 8(b)) which contains the set of company project list that is already existing in the SAP Business ByDesign system. A Projektnummer is in the format of (ProjectID – ProjectTask), where the ProjectID sits down with points and ProjectTask always specified after the hyphen, for example, 3603.001.2600 – 17. Kostentrager is used for the insurance provider and is present within their respective Projektnummer (for instance the Kostentrager 3603 in the Projektnummer 3603.001.2600 – 17). The column Aktiv remains empty if there are no foreign assignments associated with the project. Stunden refers to the total time worked by an employee on a
particular project. A single user can work on more than one projects in a month as shown in Figure 8(a).

The existing VFP system accepts input in the format of the text file; therefore, the user has to generate a text document in notepad for the corresponding Zeiterfassung. During the preparation of the text document the user copy and transfer only four main columns Personalnummer, Zeitart, Projektnummer and Stunden. The text file created in this way is the input file for the VFP conversion program.
Step 2: Initiation of the Conversion Program in VFP
The windows application of VFP conversion program is available in the local server of the company. The user can access this only within the company network. "BYDZ.PRG" conversion program used for generating the required output SAP file data. In the VFP command prompt, the program is invoked using the statement `DO BYDZ`, this begins the program and opens up the user interface.

![Figure 9(a): Loading timesheet text file in VFP user interface](image)

Step 3: Loading and Conversion of Timesheet Text File
In the user interface for the preparation of the output timesheet data file, first, the user selects the menu item named "Data preparation" under "Edit". This call for the menu item has to be performed twice that allows the user to select the text file of timesheet data, the file created in section 3.1.1(see Figure 9(a)).
Figure 9(b): Conversion of timesheet text file

The text file is then saved as XLS file format, as shown in Figure 9(b) that generates the output timesheet data in an intermediate SAP format.

**Step 4: Formatting the Generated Output SAP Timesheet Data**

The XLS file created by BYDZ program, contain new columns (columns from A to I as shown in Figure 10) similar to that of the SAP template, that includes `personalnummer`, `arbdatum`, `begin`, `ende`, `dauer`, `service`, `zeitart`, `zuschlag` and `praufgabe` for booking work time of employees in SAP Business ByDesign Application Portal. Among these fields `arbdatum`, `begin`, `ende`, `dauer` are in decimal format, the column `arbdatum` refer to the date when the employees worked whereas the other three columns represent the employees working time. The working time `begin` show the start time furthermore `ende` represents the closing time of an employee respectively. The `dauer` is the total working hours of an employee and is equivalent to the difference between `begin` and `ende` (i.e `dauer = ende - begin`).
Figure 10: Intermediate output timesheet XLS file

The decimal values of employees working time are the values similar to that in a pennyweight (refer section 2.11, Para.2). The function \textbf{VALUE (text)} in Microsoft Excel gives the exact decimal value of not only time but also constant number and date. The generated XLS file include columns with some truncated decimal values mainly for the columns from “A” to “E” w.r.t the example file sited in Figure 8. The \textbf{VALUE} function determines the exact values of all these columns and therefore copied to the nearby set of cells in the same excel sheet from “K” to “O”. Further, the entire block of \textbf{VALUE} formulas (i.e. the values in column “K” to “O”) is then copied back to the column’s “A” to “E”, since without this the data cannot be transferred to the final SAP booking template file.

**Step 5: Transferring the Data to SAP Template**

The final step is to transfer the formatted data to the SAP booking template file, that is available for download from the SAP Business ByDesign Web portal. After transporting the data to the SAP template, the excel sheet automatically converts the decimal values to its comparable formats mainly for the constant number, date, time (as presented in Figure 11). In the SAP
template, the columns “N” to “Q” must be kept empty, SAP always marks the posted data over here. Some of the praufgabe columns does not have a task number, which will automatically be determined by the SAP system after the submission of the template (notice the adjusted values marked in Figure 11).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 11: Final SAP Output Result File](image)

**Actual Posting of Timesheet**

The finally prepared SAP booking template is opened in Microsoft Excel menu item “SAP Business ByDesign” and further provide the login credentials to connect to the SAP Web portal from excel. Clicking on this item starts the actual uploading and booking of the data on to the SAP application. A window opens displaying the number of booked records when everything saved without errors. In addition to this, unique codes get filled in the column’s "N" to "Q" of SAP template indicating the successful record submission. For an erroneous row, only the column "N" of that particular record gets populated. In this case, the user copy to the excerpt
from the template that contains only the unposted records, correct the records, eliminate the existing code from the column "N", and upload that data again separately.

3.1.3 Data Analysis

The data collected based on the existing system help us to determine the input column data required for the generation of the booking data for the final SAP timesheet template. The necessary columns include Personalnummer, Zeitart, Projektnummer, Kostentrager, Aktiv and Stunden (refer section 3.1.2, Step 1). These columns will form attributes for the relational database table in the proposed timesheet management system.

The SAP booking template file contains columns mainly personalnummer, arbdatum, begin, ende, dauer, service, zeitart, zuschlag and praufgabe. The output data generated based on the input data values must be of a specific pattern and must match the final SAP template. During the distribution of cumulative work timings of each employee of the company on different projects in a particular month, there are specific rules to be followed (will be discussed in detail in section 4.1).

The IT Manager of the company has reported several errors like the false personal number, wrong task number, incorrect work time distribution etc. that SAP Business ByDesign notify during the upload of the output booking template prepared using the existing system. He says that,

“it always takes more time in fixing the error since after fixing the errors each of these corrected rows has to be copied separately into a new SAP template and then reuploaded”. - Mr.Brahim Oulmaati

Therefore, in the existing system data generated for the booking SAP template includes issues associated with the data format and also on the distribution of the work time of the employees. It is additionally understood, the uncertainty of data loss exists during the data copy and pasting by the user from the input timesheet document file (Zeiterfassung) to the text document file (the input file format needed for the VFP system). That can further result for the final cause of the error messages during the SAP template upload.

3.1.4 Result of Analysis and Interpretation

The results obtained from the analysis of the data includes:

(a) The company, GITEC – IGIP GmbH is making use of Visual FoxPro based timesheet management system, a discontinued product of Microsoft (no more version and support
The system is not fully automated and has to perform manual checking and transformation of data files during the intermediate stages of the process.

(b) Deduced that the operation speed of the system is slow having a batch of steps included for data processing. The system gets hang-up at many times, consequently making the user restart the application and repeat the steps to continue the processing.

(c) Noted that the system accepts the data file only in text format. Therefore, the user has to manually generate a text file for the corresponding input data file. The data loss during this transformation is considered to be one of the major issues in the existing timesheet management system.

(d) Manual checking and correction of the generated output result file are needed to transfer the data to the final SAP timesheet booking template. The generated output file contains truncated decimal values that should be corrected by the user before transferring the booking data to the final SAP booking template (as explained in section 3.1.2, step 4).

(e) Concluded that the final SAP booking file prepared contained issues associated with the data format. During the submission of the template on the SAP Business ByDesign web portal, the error messages generally are notified in a pop up (refer section 3.1.3, Para 3).

(f) During the investigation and data collection phase, observed that most of the company members using the system are not much aware of the workflow. They refer to the document “Hour Handover GITEC on The Example of An Hourly Ticket” (see Appendix 1) most time during the file conversion process. This study is needed to know the file path in the local server for the retrieval of the file templates used during the intermediate stages of data processing. Additionally, the complete process has multiple steps and manual effort required.

### 3.2 System Analysis

The investigation conducted on the operation of the existing Visual FoxPro based timesheet management system in the company, GITEC – IGIP GmbH show that there is a high need to speed up the process of updating the working hour details of the employee's order for the monthly salary calculation by the finance department. The company spend a lot of hours in the process of generating the correct timesheet booking template data for updating the SAP Business ByDesign system.
In the existing system, the manual approach of copy-pasting of information at intermediate stages of processing hampers the data transfer speed of the system. Incidents of a manual copying error of records have also reported which leads to the generation of erroneous output booking data for the SAP template. The user must check the intermediate result generated between different stages to ensure the final result to be correct. They need to memorize the necessary process steps in the system and further if the application system stops in between they must restart it again to continue.

### 3.2.1 Problems of Existing System

From the system analysis made above, the issues identified in the current operating system are:

1. There is a lot of delay in processing due to the intermediate hanging of the Visual FoxPro application system.
2. There are noticeable hours spend on processing the timesheet input data file and release of the final booking data for the SAP system.
3. The finance department finds it difficult in monthly salary calculation due to the late updating in SAP Business ByDesign web portal.
4. The user interface of the system needs more improvement and must support multiple platforms (currently the existing system operates only on Windows operating systems).
5. The developers in the IT department finds it difficult for the system maintenance due to lack of support of the outdated version of the existing system developed on Visual FoxPro.
6. The users find difficulty in several stages of the timesheet processing that involves manual copy-pasting of data.
7. There are constant issues regarding the data format associated with the final booking SAP template created.
8. There is prolonged difficulty in decision making about the accuracy of data file generated since SAP Business ByDesign web portal lists the erroneous record list only after the submission of the resultant booking data file.

### 3.2.2 Proposal of a New System

The proposed system is an intranet that is available for the company employees and any authorized persons within the company. They can access the system through any web browser like Firefox, Google etc. within the company network. The system has a responsive, user-
friendly interface and supports multiple platforms like smartphones, desktop computers, iPad etc. The proposed system bypasses all the numerous stages of processing in the existing system and enables the users to attain the final result in a single step, by uploading the input timesheet file to the system. The final result generated is the SAP template file embedded with booking data.

3.2.3 Benefits of The Proposed System

The new system implemented within the company has enhanced the timesheet management process in the following manner:

1. Efficiency in operation with less manual data transfer and changes. The single step mechanism for output generation improves the operational speed.
2. User-friendly and responsive web application interface with online support using any web browser.
3. Faster data transformation and processing by the integration of SQL Stored Procedures.
4. Data format checking using the server-side code in .NET and Spreadsheet light library for .NET Framework written in C#.
5. Remove the processing delay and accelerates the timesheet management processing mechanism with improved timeliness and accuracy.

4 System Design and Implementation

This chapter will describe the proposed web application system design and its implementation based on the Model-Driven Development (MDD) methodology. Firstly, in the design phase, the system requirements are examined furthermore we describe the MVC(Model-View-Controller) design pattern and the 3 – tier architecture implementation in ASP.NET with EF (Entity Framework) followed by the preparation phase that details about the EF – Code First Approach. Lastly, the implementation phase that includes the server-side C# coding along with the SQL database stored procedure and associated functions to process the timesheet data and generate the final SAP timesheet booking data file.

4.1 Design Phase

The initial phase of the Model-Driven Development process (refer section 2.3) focus on identifying the customer requirement that the application needs to fulfil. That mainly includes understanding the functional and non-functional requirements of the customer.
4.1.1 Functional Requirements

Functional requirements capture the intended behavior of the system and determine the necessary task, action or activity, that must be carried out by the system (Malan and Bredemeyer 2001, p.1). They refer to the application features that must be implemented by the developers to enable the users to accomplish their tasks (Altexsoft.com, 2018).

Table 2: Basic Application Functionalities

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
<th>ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>Allow the registered employees to sign in.</td>
<td>Admin, Employee.</td>
</tr>
<tr>
<td>Add new employee</td>
<td>Add a new employee to the system.</td>
<td>Admin.</td>
</tr>
<tr>
<td>Edit profile details</td>
<td>Update existing employee profile details.</td>
<td>Admin, Employee.</td>
</tr>
<tr>
<td>Manage user permission</td>
<td>Set Add/Edit/Delete permission.</td>
<td>Admin.</td>
</tr>
<tr>
<td>Generate Timesheet file</td>
<td>Upload and generate SAP timesheet booking file.</td>
<td>Admin, Employee.</td>
</tr>
</tbody>
</table>

The significant application functionalities (see Table 2), the login page allows the registered employees to access the web application portal after their successful authentication using their username and password. The employees having an admin role can create a new employee account. All the registered employees can change their profile information after logging into the system. Admin can only grant permission of Add/Edit/Delete access to the registered users within the application system. Both admin and employees with necessary permission can upload and generate the SAP booking timesheet file.

4.1.2 Non-Functional Requirements

Chung and Cesar Sampaio do Prado Leite (2009: p.365) define non-functional requirement as, requirements which are not particularly concerned with the functionality of a system and place restrictions on the product under development and the development process, and they specify external constraints that the product must meet. They are usually the non-behavioral aspects of the system or the features that are needed but not essential to be functioning. The critical non-functional requirements recognized for the proposed system are:

A. Follow object-oriented concepts and use MVC framework (refer section 2.4) for faster extensibility and maintenance.

B. Adhere to 3 – tier architecture implementation in ASP.NET with EF (Entity Framework).
C. Use Microsoft Visual Studio IDE for web application development and Microsoft SQL Server relational database for data storage (based on the requirement from the company hosting provider).

D. Comply with latest web standards; HTML5, CSS3, Bootstrap and support recent versions of Chrome, Firefox, Safari, IE, and Opera.

E. Client-side input verification through JavaScript or jQuery.

F. Responsive user interface design using bootstrap.

G. SQL Server stored procedure and associated user-defined functions used for faster data manipulation.

4.1.3 Use Cases

The functional requirements are depicted using use cases, especially for object-oriented systems. Without considering the system internals, the use case capture who (actor) does what (interaction) with the system for what purpose (goal) (Malan and Bredemeyer 2001, p.2). The use case diagrams are the pictorial representations of the use cases and show a high-level overview of the relationship between actors, different use cases and the system (Altexsoft.com, 2018). The following Table 3 describes the elements in a use case diagram mainly used for the proposed web application system (see Figure 12).

Table 3: Use case diagram elements and description (Docs.microsoft.com, 2018)

<table>
<thead>
<tr>
<th>ID</th>
<th>ELEMENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| A | Actor | Represents a user or an external system that interacts with the application.  
A.1: Employee.  
A.2: Administrator (or Admin). |
| B | Use Case | Represents the actions carried out by one or more actors to accomplish a particular goal.  
B.1: Login.  
B.2: Update profile details.  
B.4: Generate SAP booking file.  
B.5: Download SAP booking file.  
B.6: Add new employee.  
B.7: Manage user permission. |
| C | Association | Specifies the involvement of an actor in a use case. |
| D | Subsystem | The application or system under consideration (for instance the proposed timesheet management web portal). The use cases supported |

32
by the system are present within the rectangle.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Include</td>
</tr>
<tr>
<td>F</td>
<td>Inheritance</td>
</tr>
</tbody>
</table>

Figure 12: Use Case Diagram

The use case diagram for the proposed web application portal shown in Figure 12, with two actors Employee and Administrator (or Admin). Admin is the specialized actor since it inherits the use cases, attributes and associations of its generalized element (i.e. Employee). It indicates that the Admin is a company Employee who can perform all the functions of an Employee and also have more additional functionalities or use cases like B6 and B7 respectively (refer Table 3).
4.1.4 Activity Chart

After the specifying the actors and the use cases with each actor, we provide a set of activity charts (also referred as activity diagrams) to describe main use cases in the use case diagram (see Figure 12). Activity charts are graphs connecting states using arrows referred to as transitions. The two main elements comprise of the initial state (starting point of activity chart) and the final state (ending point of activity chart). Transitions are conditions/actions, representing the condition to be examined and the actions to be attained for a state change (Almendros-Jiménez and Iribarne 2005, p.146). Alternative paths are represented using diamonds between the transitions depending on particular boolean condition.

The main activity chart of the use case for generating the SAP booking template is in Figure 13. It shows how the user after successfully logging into the system, initially selects the sub tab for uploading the input timesheet data file for a particular month. If the timesheet data template submitted is a valid one then all the records in the uploaded timesheet file, Zieterfassung (see Figure 8(a)) inserted into the SQL server database table. After the insertion process, the system calls the user-defined Stored Procedure (SP) residing within the database. The SP processes the input timesheet data and generates the result and further send back to the system that updates the processed data in the SAP template format and creates the final SAP booking excel spreadsheet file.
4.1.5 Three Tier Architecture Design

The server-side coding and web application development are carried out in Microsoft Visual Studio IDE. The first step to create web application involves the implementation of a three-tier architecture in ASP.NET MVC with Entity Framework. A typical ASP.NET MVC application comprises of Models, Views and Controllers (refer section 2.4).
Figure 14: Three-tier architecture

Figure 14, depicts the 3-tier architecture implemented in the web application system and the main layers are,

1. **Presentation Logic Layer**

The Presentation Logic Layer comprises of the web application project with MVC design pattern.

   a) **View**: The User Interface (UI) includes all the form control elements like textboxes, buttons, labels etc. These ASP.NET web pages use the Razor syntax. Razor is a server-side markup language that embeds server-side code into web pages using C# and ASP.NET. In an ASP.NET MVC project, the view engine enables to use of Razor in the application to generate HTML([www.tutorialspoint.com](http://www.tutorialspoint.com), 2018). A simple C# Razor syntax looks like @VariableName, where VariableName is the server-side variable that contains some value. We mainly associate each view page in the application with a ViewModel and to use them in the view we use @ViewModel (consider the UserLoginViewModel in Login View page: @model GITEC_TimeSheet.DTO.ViewModel.UserLoginViewModel). HTMLHelper class are used to generate HTML elements using the ViewModel class object associated with the razor view page. The commonly used HTMLHelper methods and corresponding HTML control used within the web application mentioned in Table 4.
Consider an example for the HTML Helper – Drop Down List used in the application:

```csharp
@Html.DropDownListFor(model => model.DepartmentID, 
    new SelectList((System.Collections.IEnumerable)ViewData["UserDepartmentList"], "Value", "Text"), "Select", htmlAttributes: new { @class = "form-control" })
```

Table 4: HTML Helper Methods and HTML Controls

<table>
<thead>
<tr>
<th>HTML HELPER</th>
<th>HTML CONTROL</th>
<th>SYNTAX OF HELPERS IN RAZOR VIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Html.ActionLink</td>
<td>Anchor link</td>
<td>@Html.ActionLink(title, action method, controller name)</td>
</tr>
<tr>
<td>Html.TextBox</td>
<td>Textbox</td>
<td>@Html.TextBoxFor(expression, htmlAttributes object)</td>
</tr>
<tr>
<td>Html.CheckBox</td>
<td>Checkbox</td>
<td>@Html.CheckBoxFor (expression, htmlAttributes object)</td>
</tr>
<tr>
<td>Html.HiddenFor</td>
<td>Hidden field</td>
<td>@Html.HiddenFor (expression, htmlAttributes object)</td>
</tr>
<tr>
<td>Html.PasswordFor</td>
<td>Password textbox</td>
<td>@Html.PasswordFor (expression, htmlAttributes object)</td>
</tr>
<tr>
<td>Html.DisplayFor</td>
<td>Html text</td>
<td>@Html.DisplayFor (expression)</td>
</tr>
<tr>
<td>Html.LabelFor</td>
<td>Label</td>
<td>@Html.LabelFor (expression, htmlAttributes object)</td>
</tr>
<tr>
<td>Html.DropDownList</td>
<td>Dropdown</td>
<td>@Html.DropDownListFor(expression, select list items, string optional label, htmlAttributes object)</td>
</tr>
</tbody>
</table>

Data validation in the application is handled using the DataAnnotation attribute that contains built-in validation attributes associated with different validation rules. These rules applied to the ViewModel classes of a particular view page and automatically enforce these rules and display validation message in the view (Tutorialsteacher.com, 2018). Consider the example below for the attribute Username:

**View Element**

```csharp
@Html.ValidationMessageFor(model => model.Username, null, 
    new { @class = "mandatory" })
```

**ViewModel Attribute**

```csharp
// Gets or sets the Username (of UserLoginViewModel) having data annotation validator

[Required(ErrorMessage = "Please enter Username")]
public string Username { get; set; }
```

b) **Model**: In the application system, each database table represented in the form of a .cs class file. All views are associated with a model object still all information of the object not required on a Razor view page (user interface). Therefore, instead of using the model,
we create a View Model that aggregate required pieces of information for a particular view. View Model's further support data annotation and validation that is not a part of its corresponding domain specific model.

c) Controller: They are .cs class files responsible for handling the HTTP requests, retrieving the data model and passing the model to the respective view.

2. Business Logic Layer
Before storing the data to the database or after reading the data from the database, some logical operations performed on the data in this layer using C#.NET programming language.

3. Data Access Layer
The system database is MS SQL and to access the data from the database we are using Entity Framework Code First mechanism. The following C# code is used to fetch the record from the tables in SQL server database using GITEC_TimeSheetContext (refer section 2.1).

```csharp
// The data repository
public class Repository<TEntity> : IRepository<TEntity> where TEntity : class
{
    // the context
    private GITEC_TimeSheetContext context;
    // the entity set
    private DbSet<TEntity> entitySet;
    // Initializes a new instance of the Repository<TEntity> class
    // <param name="context">the context</param>
    public Repository(GITEC_TimeSheetContext context)
    {
        this.context = context;
        this.entitySet = context.Set<TEntity>();
    }
    // Gets the entities as a query for further manipulation
    // <param name="filter">where clause</param>
    // <param name="includeProperties">include properties</param>
    public IQueryable<TEntity> GetQuery(
        Expression<Func<TEntity, bool>> filter,
        string includeProperties = ""
    )
    {
        int total;
        return GetQuery(out total, filter, null, false, 0, 0, includeProperties);
    }
    // Gets the entities as query for further manipulation
    // <param name="totalCount">returns total count of entities in the result set</param>
    // <param name="filter">where clause</param>
    // <param name="orderBy">order by clause</param>
    // <param name="enablePaging">true if paging must be</param>
    // <param name="countPerPage" value="10">number of entities per page</param>
    // <param name="includeProperties">include properties</param>
}```
matching entities as a query

```csharp
public IQueryable<TEntity> GetQuery(
    out int totalCount,
    Expression<Func<TEntity, bool>> filter = null,
    Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>> orderBy = null,
    bool enablePaging = false,
    int curPage = 0,
    int countPerPage = 10,
    string includeProperties = ""
)
{
    IQueryable<TEntity> query = this.entitySet;

    if (filter != null)
    {
        query = query.Where(filter);
    }
    foreach (var includeProperty in includeProperties.Split(new char[] { ',', ' ' }, StringSplitOptions.RemoveEmptyEntries))
    {
        query = query.Include(includeProperty);
    }
    totalCount = query.Count();
    if (orderBy != null)
    {
        if (enablePaging)
        {
            return orderBy(query).Skip(curPage * countPerPage).Take(countPerPage);
        }
        else
        {
            return orderBy(query);
        }
    }
    else
    {
        return query;
    }
}
```

Business Object

Transfer of data from one layer to another is carried out by the business object. Business Object (BO) is also referred to as a Data Transfer Object (DTO) act as a container that stores the data and passed to other layers. All the ViewModel classes stored in this layer.

4.1.6 Database Design

In the project we use Entity Framework (EF) Code-First approach to design the database initially, coding is done C#.NET and then EF further create the database from the code. Here in this technique, the developer creates classes for the domain entity and with the help of code-based migration creates or updates the database schema in SQL server. The Entity-Relationship diagram of system database (GITEC_TimeSheet) shown in Figure 15 and all table represents the
domain classes generated from the server-side C# coding. The additional table _MigrationHistory is generated by EF, to keep track of the migrations or the modifications done on the table.

![Entity Relationship Diagram](image)

**Figure 15: SQL Server Entity Relationship Diagram**

### 4.2 Preparation Phase

The initial phase focuses on setting up the EF development environment in Visual Studio IDE and then using EF Code First Migration for creating and updating the database schema.

#### 4.2.1 Entity Framework – Environment Setup

To develop an Entity Framework based web application system, we install the following development tools:

a) Visual Studio 2013 or above.

b) SQL Server 2012 or above.

c) Entity Framework updates from NuGet\(^3\) Packages.

---

\(^3\) NuGet is an open source Visual Studio extension for tools and service.
NuGet Packages accomplish the installation of EntityFramework.dll (EF API service) in MVC web application project in Visual Studio.

4.2.2 Entity Framework – Code First Migration

After the successful installation of EntityFramework API in Visual Studio, we can use the Code First Migration mechanism for the creation of database tables corresponding to the defined domain classes in the web application project.

A. Database Initialization in Entity Framework: EF determines the database name and server during database initialisation using the Code First approach. The connection between the web application project in Visual Studio IDE and SQL Server is established using the connectionStrings in the web.config file in the application, having the provider name specified as “System.Data.SqlClient”.

Web.config:

```xml
<configuration>
    <connectionStrings>
        <add name="GITEC_TimeSheetContext" connectionString="data source=GT-LT_34\SQLEXPRESS;initial catalog=GITEC_TimeSheet;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"
             providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

GITEC_TimeSheetContext.cs

```csharp
namespace GITEC_TimeSheet.DAL
{
    public partial class GITEC_TimeSheetContext : DbContext
    {
        public GITEC_TimeSheetContext() : base("name=GITEC_TimeSheetContext")
        {
        }
    }
}
```

The base constructor with the parameter "name=GITEC_TimeSheetContext" will create the database “GITEC_TimeSheet” (is the initial catalogue name in connection string) after migration.

B. Configure Domain Class using EF Fluent API: The Domain classes are created and specified within the GITEC_TimeSheetContext.cs file. OnModelCreating of DbContext is used to override and configure the domain class while building a model. DbModelBuilder is referred to as the fluent API as it calls multiple methods in a sequence.
namespace GITEC_TimeSheet.DAL
{
    public partial class GITEC_TimeSheetContext : DbContext
    {
        public GITEC_TimeSheetContext ()
            : base("name=GITEC_TimeSheetContext")
        {
        }

        public DbSet<Users> Users { get; set; }
        public DbSet<Zeiterfassung> Zeiterfassung { get; set; }
        public DbSet<UserRole> UserRole { get; set; }
        public DbSet<Department> Department { get; set; }

        protected override void OnModelCreating (DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Users> ();
            modelBuilder.Entity<Zeiterfassung> ();
            modelBuilder.Entity< UserRole> ();
            modelBuilder.Entity< Department> ();
        }
    }
}

C. Code – Based Migration: Code-base migration allows us to create the database with all specified classes as tables in SQL server database. The following commands executed in the Package Manager Console in Visual Studio:

1. Enables-Migrations: Enables migration in the project.
2. Add-Migrations: Generates the migration class for the tables to be created in the database.
3. Update-Database: Finally, executes the migration file generated plus updates the changes to the database schema.

4.2.3 Bootstrap Template and Theme Integration

After a successful setup of the web application project for server-side coding and creating and establishing the connection with the application using EF Code First mechanism, the next major step is the integration of open source AdminLTE template and theme built on top of Bootstrap 3. It provides a range of responsive and reusable dashboard and control panel theme.

Essential steps of integrating the downloaded AdminLTE template into our ASP.NET MVC project are:
1) Download a Theme: First step is to have a basic bootstrap theme for applying into the project. The AdminLTE theme is available in the https://adminlte.io/docs/2.4/installation for free download and with documentation (used in this project). The downloaded theme comprises a set of folders with cascading style sheets (CSS), images, java scripts (JS), plugins and the HTML web pages. Open the web page named as index.html in a text editor and copy the content to clipboard.

2) Add Theme HTML: Paste the content of the HTML theme page to _Layout.cshtml file in our ASP.NET web application.

3) Add Theme Resource Files: There are several folders possessing resource files (as mentioned in step 1) used in the integrated HTML theme. In AdminLTE theme, the folders are bower_component and dist. Add these two folders into our application.

4) Formatting the Folder Path: The path reference of the files must be corrected accordingly for the new integrated HTML theme according to the folder path in the web application. The content section of the HTML template must contain “@RenderBody()” that acts as a placeholder for the application system to determine the place to render the contents of the view using the layout page.

4.3 Implementation Phase

After investigating the client (in our project GITEC-IGIP GmbH) requirement, setting up the development environment and implementing the database design, a selected number of use cases are chosen to describe the scenarios along with the C#.NET code and SQL Stored Procedure (SP).

4.3.1 Login

Table 5: Use Case 1 - Login

<table>
<thead>
<tr>
<th>Title</th>
<th>Account Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Admin, Employee</td>
</tr>
<tr>
<td>Scenario</td>
<td>1. System request actor to enter login credentials.</td>
</tr>
<tr>
<td></td>
<td>2. Actor enter required login credentials.</td>
</tr>
<tr>
<td></td>
<td>3. System validates the entered credentials and redirect the actor to the main page.</td>
</tr>
</tbody>
</table>
The Model

The two central users of the system are the administrator and the employees of the company. The user inputs their username and password to access the web application portal. The model class used here is *Users* (see Appendix 3) and correspond to the SQL Server database table *Users* (see Figure 15).

![Sign in to start your session](image)

**Figure 16: Login Screen**

The ViewModel

For the login view page, all the attributes of the model class not required. Therefore, we model a view model class for the login page, *UserLoginViewModel* (see Appendix 4) holding the necessary field attributes such as *username* and *password* and controls the validation on the login page utilizing model validation (refer section 4.1.5).

The View

The login view page (see Appendix 5) uses the AJAX form submission method using the view model *UserLoginViewModel*. Submission of form by the user clicking on the *Sign In* button, the values entered posted to the *LoginDetails* function in *Login* controller for processing.

The Controller

The *LoginController* (see Appendix 6) construct mainly has two functions for GET and POST request handling. The GET method initializes the view model object *UserLoginViewModel* for the corresponding Login view page whereas the POST method accepts the submitted form attribute values for processing. Based on the processed result if the user details are valid, then the
control is redirected to the main page of the web site. Otherwise, the user prompted with a message box showing invalid login credentials.

**BLL Function**

The function *Authenticate* written in BLL is invoked from the POST method for the authentication check. The function accepts the parameters username and password provided by the user from the view or the user interface and check it against the database records via the DAL layer. If the user exists, then *Authenticate* function fetches essential attribute values of the user from the database table *Users* (refer Figure 15) and returns to the controller.

```csharp
// Function to authenticate the user
// <returns>returns User Details</returns>
public UserLoginViewModel Authenticate (string userName, string password)
{
    // View model used to fetch the user details if the posted parameters are found in database.
    UserLoginViewModel userLoginModel = new UserLoginViewModel();

    // Model initialization from database.
    Users userLoginObj = this.unitOfWork.UsersRepository.GetQuery (u => u.Username == userName && u.Password == password).FirstOrDefault();

    // In case if model object is null then not valid else populate the ViewModel.
    if (userLoginObj! = null)
    {
        userLoginModel.FirstName = userLoginObj.FirstName;
        userLoginModel.LastName = userLoginObj.LastName;
        userLoginModel.UserID = userLoginObj.UserID;
        userLoginModel.UserRoleID = userLoginObj.UserRoleID;
        userLoginModel.UserRoleCode = userLoginObj.UserRole.UserRoleName;
        userLoginModel.ProfileImagePath = userLoginObj.ProfileImagePath;
    }
    return userLoginModel;
}
```

### 4.3.2 Add New Employee

The admin user adds a new employee in the web application system by entering the basic profile details and assigning the user role and the department.

**Table 6: Use case 2 – Add New Employee**

<table>
<thead>
<tr>
<th>Title</th>
<th>Add New Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Admin</td>
</tr>
<tr>
<td>Scenario</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Admin actor enter the basic information of the new user.</td>
</tr>
<tr>
<td></td>
<td>2. System provide dropdown lists on the form to assign the role and department of the new user.</td>
</tr>
</tbody>
</table>
3. System offer the actor to add a new user.

The ViewModel
The model class associated in this scenario remain the same whereas the view model associated with it is EmployeeVM (see Appendix 7) with required field attributes.

The View
The AJAX form submits on save button click on the EmployeeDetails view page (see Appendix 8) and post the field values via EmployeeVM to the SaveEmployeeDetails function in Employee controller for processing.

![Add New Employee View](image)

Figure 17: Add New Employee View

The Controller
The HttpGet request for the function AddEmployee (refer Appendix 9 for the implementation) is used to initialize the view model EmployeeVM and also to populate the dropdown list in the view page with values from the database. The submitted view form details by the user are saved to the database by the SaveEmployeeDetails HttpPost method (see Appendix 9). The processing of the data is performed by the SaveEmployee method in BLL and based on the processed result the user prompted with a message box showing the corresponding status message.
**BLL Function**

The function *SaveEmployee* in BLL initially checks whether any identical user exists in the system based on the email address provided. Depending on this the *EmployeeExistStatus* attribute in *EmployeeVM* is updated. If the user is new, then a random password is generated using the user-defined function *GenerateRandomPassword* and update the view model attribute field for password. After the successful updating of the database via the DAL layer, a confirmation mail containing the user credentials sent to the respective user.

```csharp
/// <summary>
/// Saves Employee
/// <summary>
/// <param name="model">Employee view model</param>
/// <returns>Updated employee view model</returns>
public EmployeeVM SaveEmployee(EmployeeVM model)
{
    var userInstance = this.GetUserDetailsByEmail(model.EmailAddress);
    if (userInstance != null)
    {
        model.EmployeeExistStatus = 0; // Already registered
        return model;
    }

    Users obj = new Users();
    obj.CreatedDate = DateTime.Now;
    obj.CreatedByID = 1;
    obj.Username = model.EmailAddress;
    obj.Password = this.GenerateRandomPassword(model.EmailAddress);
    obj.FirstName = model.FirstName;
    obj.LastName = model.LastName;
    obj.EmailAddress = model.EmailAddress;
    obj.PersonalNummmer = model.PersonalNummmer;
    obj.DOB = model.DOB;
    obj.StundennachweisPath = model.StundennachweisPath;
    obj.UserRoleID = model.UserRoleID;
    obj.DepartmentID = model.DepartmentID;
    obj.UserID = 0;
    obj.ModifiedByID = 1;
    obj.ModifiedDate = DateTime.Now;

    obj = this.SaveEmployee(obj);
    if (model.UserID == 0)
```
4.3.3 Generate SAP Booking File

After the completion of two significant use cases mentioned in the above sections, the users can now access the web application system for generating the SAP booking file. This use case shows the core processing mechanism of this thesis, where the user can upload the monthly timesheet document directly to the system through the web application interface and the SQL stored procedures and its associated scalar valued function process the data and automatically generates the SAP booking file using the server side SpreadsheetLight library (refer section 2.11).

Figure 18: SAP Booking Process
The user initially downloads the timesheet booking file template (see Figure 8(a)) in Generate SAP File section in the web application. The Finance department records the cumulative time record details of the employees in the blank template file. The file then uploaded to the web application system furthermore the system performs the data processing and generates the required SAP booking file. The file data further transferred to the SAP Business ByDesign Interface with the help of Microsoft Excel (MS Excel) SAP Add-in. The SAP user credentials are provided to link MS Excel to SAP account portal and once after establishing the connection user can successfully update the timesheet details of the employees by using the system generated SAP booking file (see Figure 22).

Table 7: Use case 3 – Generate SAP Booking File

<table>
<thead>
<tr>
<th>Title</th>
<th>Generate SAP Booking File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Admin, Employee</td>
</tr>
<tr>
<td>Scenario</td>
<td>1. Actor downloads the timesheet document template.</td>
</tr>
<tr>
<td></td>
<td>2. Actor uploads the monthly timesheet document.</td>
</tr>
<tr>
<td></td>
<td>3. The system generates SAP booking template for the uploaded timesheet.</td>
</tr>
<tr>
<td></td>
<td>4. Actor selects specific month and year from the dropdown to generate SAP booking file.</td>
</tr>
<tr>
<td>Alternative</td>
<td>1. When user upload timesheet document for an existing period, then the system asks whether to overwrite.</td>
</tr>
</tbody>
</table>
The model class used here is Zeiterfassung (see Appendix 10) and correspond to the SQL Server database table Zeiterfassung (refer Figure 15). Each attribute names in the class models the main column field names present in the timesheet document.
The ViewModel

The view model associated with the tabbed view pages for timesheet document upload (see Figure 23) is `UploadZeiterfassungVM` and generate SAP file (see Figure 24) is `SAPPageVM` respectively. `UploadZeiterfassungVM` includes all the attributes from the `Zeiterfassung` model class along with `ZeiterfassungDoc` attribute of `HttpPostedFileBase` data type that supports file upload.

```csharp
public class SAPPageVM
{
    /// <summary>
    /// Gets or sets the Month
    /// </summary>
    [Required(ErrorMessage = "Please Select Month")]
    [DisplayName("Month")]
    public int Month { get; set; }

    /// <summary>
    /// Gets or sets the Year
    /// </summary>
    [Required(ErrorMessage = "Please Select Year")]
    [DisplayName("Year")]
    public int Year { get; set; }
}
```

The View

The central difference among the tabbed view pages web interface for generating the SAP booking file is that,

1. Upload Zeiterfassung Tab View: The user uploads the Zeiterfassung document and selects the month and year of employee timesheet detail collected in the dropdowns according to the Zeiterfassung document for a new entry. The view posts the data submitted by the user to the `UploadZeiterfassung` function written in the `Admin` controller. The JavaScript on the view page checks the error status in the status container `TempData["SuccessAlert"]`. The two main status validations include:

   a. The system tells the user that the file extension type is wrong and it must be similar to the .xlsx file format (i.e. the file must be equivalent to the Zeiterfassung template format provided on the page).

   b. If the user submits for an already existing period, then the system generates an alert message box asking the user whether to overwrite an existing timesheet data (shown in Appendix 11).
2. Generate SAP File Tab View: The user selects the month and year for an already existing SAP file detail in the system. The view transmits the data selected by the user to the GenerateSAPTemplate function written in the Admin controller. The razor view code for the Generate SAP File Tab View shown in Appendix 12.

The Controller

The UploadZeiterfassung HttpPost method in the AdminController accepts the uploaded timesheet data file and transfer it to the SaveZeiterfassung method in BLL and on the other hand the GenerateSAPTemplate method provides the default SAP booking template file path from the server and the details submitted by the user to the SAPFileGenerate function (see Appendix 13) written in BLL for further processing.

```csharp
public class AdminController : Controller
{
    // POST: Generate SAP Template HTTP POST Method
    [HttpPost]
    public ActionResult GenerateSAPTemplate(SAPPageVM details)
    {
        var CurrentDate = DateTime.Now;
        string fName = "SAPTemplate.xlsx";
        string tempFolderName = ConfigurationManager.AppSettings["GeneratedSAPTemplate"];
        string tempFolderPath = Server.MapPath("~/" + tempFolderName);

        var fileUrl = Path.Combine(tempFolderPath, fName);
        var result = this._manageEmployee.SAPFileGenerate(fileUrl, details);

        tempFolderPath = "~/UserData/SAPTemplate/" + fName;
        fileUrl = fileUrl.Replace("SAPTemplate", "UploadedSAPFiles");
        return File(fileUrl, "application/vnd.ms-excel", fName);
    }
}
```

BLL Function

The function SaveZeiterfassung in BLL save the detail within the uploaded timesheet document to the SQL server database table Zeiterfassung by establishing a connection through the InsertZeiterfassungRecordToDb method. After the successful updating of the database SAPFileGenerate method make use of the SLDocument object sl commonly used in SpreadsheetLight (refer section 2.11) to generate final SAP spreadsheet booking file by fetching the processed timesheet data from the database by using the stored procedure.

**Stored Procedure and User-Defined Functions**

The cumulative timesheet data for a specific month once saved to the database the system call the SQL Server Stored Procedure ‘generateSAPTemplate’. The SP then accepts the two parameters month and year selected by the user from the corresponding view page for the selection of the cumulative timesheet records for the period from the Zeiterfassung table. The fetching of all input records or the row details from the Zeiterfassung table done by a table variable @Zeiterfassung_TempTbl with similar column filed declared whereas the final output data stored in different table variable @SAPTemplateFinal that contains the column names needed for the final SAP booking file template. The full implementation code for SP generateSAPTemplate presented in Appendix 14.

```sql
USE [GITEC_TimeSheet]
GO
// Object:  StoredProcedure [dbo].[generateSAPTemplate] to generate the sap file data.
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[generateSAPTemplate] @month int = NULL, @year int = NULL
AS
BEGIN

DECLARE @counter INT = 1,
        @max INT = 0,
        @mdatum datetime,
        @mperson nvarchar(100)= NULL,
        @mersttag datetime,
        @mstunden int = NULL,
        @mservice nvarchar(100)= NULL,
        @mprojekt nvarchar(100)= NULL,
        @mende nvarchar(100)= NULL,
        @mbegin nvarchar(100)= NULL,
        @mpending nvarchar(100)= NULL,
        @mduration nvarchar(100)= NULL,
        @tempstunden nvarchar(100)= NULL,
        @currentID int = NULL,
        @previousID int = NULL

    //Container to fetch details from database table
    DECLARE @Zeiterfassung_TempTbl TABLE (ID int identity,
                                             ...
```

//Container to store the final result for populating the SAP booking template.

DECLARE @SAPTemplateFinal TABLE (  [ID] int,  personal nvarchar(100),  arbdatum nvarchar(100),  arbdatumSAP nvarchar(100),  beginn nvarchar(100),  ende nvarchar(100),  dauer nvarchar(100),  [service] nvarchar(100),  praufgabe nvarchar(100),  mersttag datetime,  pending int)

//Insert your required data in the variable of type TABLE.
INSERT INTO @Zeiterfassung_TempTbl
SELECT [ZeiterfassungID], [PersonalNummmer], [Stunden], [Kostenträger], [Month], [Year]
FROM [dbo].[Zeiterfassung] where Month = @month and Year = @year

//Initialize the @max variable. We’ll use this variable in the next WHILE loop.
SELECT @max = COUNT([ID]) FROM @Zeiterfassung_TempTbl

//Each record in the input timesheet file stored in Zeiterfassung table fetch.

WHILE @counter <= @max
BEGIN

    // Code region for initializing the basic start and end time based on the criteria mentioned in following sub sections/

    //SP Call to SAPFormatGenerator function where each function is applied.
    SELECT @mende=ende, @mdatum= mersttag, @mduration = dauer, @mpending = pending
    FROM dbo.SAPFormatGenerator(@mperson, @mdatum, @mstunden, @mservice, @mprojekt, @mbegin, @mpending) ti

    //Increment WHILE loop @counter variable and it act as the row pointer.
    SET @counter = @counter + 1
END

SELECT * FROM @SAPTemplateFinal
The stored procedure recursively invokes the table-valued function (refer section 2.8: para. 4), 
\{dbo\}.\{SAPFormatGenerator\} by passing each row information present in the 
@Zeiterfassung_TempTbl variable. The full implementation code for the table-valued function 
SAPFormatGenerator given in Appendix 15. The function performs the splitting of the 
cumulative working time of each employee (i.e. each row information) and returns the result in 
the format of a table that is similar to a @SAPTemplateFinal variable. The distribution of the total 
working time of the employees on specific projects based on the following criteria:

A. Business hours for all the company employees in a regular workday range between 08:00 
to 16:18 (in Hour: Minute format).

//Initial time for business hour setting in SP

SET @mbegin = '08:00:00'

IF( @mende = '16:18:00')

BEGIN
  SET @mbegin = '08:00:00'
END

ELSE
  BEGIN
    SET @mpending = @mpending
    SET @mbegin = @mende
    SET @mdatum = @mdatum
  END

B. The maximum duration must be 08:18 hours, that is the difference between the begin and 
end hour of a business day.

//Set up the total duration as the difference between start and end hour and type conversion

DECLARE @msrest int = 498, 
          @mmstarttag int = 0, 
          @mmendtag int = 0, 
          @mdauer int = 0, 
          @msdauer int= 0, 
          @mmdauer int= 0

SET @mmendtag =16*60+18
SET @mmstarttag = @mmendtag - @mpending

SET @mdauer = @mmendtag - @mmstarttag
SET @msdauer = Convert(int, @mdauer/60)
SET @mmdauer = @mdauer - @msdauer*60
C. If an employee has worked in multiple projects, then the assignment of the time is managed by seeing the closing hour of the employee for the former project furthermore by updating the beginning hour with the end hour (only at the first instance).

    SET @mmstarttag = @mmendtag - @mpending

D. The work hours of an employee not to be allotted for the weekend. During data processing the user-defined scalar-valued function [dbo].[IsWeekend] invoked within the table-valued function accepts a date argument and determines whether it indicates the weekend.

    //User-defined scalar function IsWeekend to check the date is weekend or not

    CREATE FUNCTION [dbo].[IsWeekend](@mdate datetime = NULL)
    RETURNS int
    AS BEGIN
        DECLARE @mret int = 0
        IF DATEPART(WEEKDAY, @mdate) = 1 OR DATEPART(WEEKDAY, @mdate) = 7
        BEGIN
            SET @mret = 1
        END
    RETURN @mret
    END

E. Apart from the weekend, work hours of an employee not to be assigned on holidays in a year, that is calculated by using the scalar-valued function [dbo].[IsHoliday] invoked within [dbo].[SAPFormatGenerator]. A scalar-valued function [dbo].[IsEaster] (code presented in Appendix 16) is invoked within the function [dbo].[IsHoliday] to determine the Easter holiday in a particular year. The function [dbo].[IsHoliday] accepts a date argument and returns an integer value 1 if holiday and 0 otherwise.

    //User-defined scalar function IsHoliday to check the date is public holiday or not

    CREATE FUNCTION [dbo].[IsHoliday](@mdate datetime = NULL)
    RETURNS int
    AS BEGIN
        DECLARE @mret int = 0,
                @mostern int,
                @mdatenum int,
                @year int
        SELECT @mostern = dbo.IsEaster(DATEPART(YEAR, @mdate));
SET @mdatenum =
DATEDIFF(DAY,DATEFROMPARTS(1900,1,1),DATEFROMPARTS(DATEPART(YEAR, @mdate),DATEPART(MONTH, @mdate),DATEPART(DAY, @mdate))) + 2

SELECT @mret = CASE WHEN DATEPART(DAY, @mdate) = 1 AND DATEPART(MONTH, @mdate) = 5 THEN 1 – New Year
WHEN DATEPART(DAY, @mdate) = 1 AND DATEPART(MONTH, @mdate) = 5 THEN 1 – May 1
WHEN DATEPART(DAY, @mdate) = 3 AND DATEPART(MONTH, @mdate) = 10 THEN 1 – Day of German Unity
WHEN DATEPART(DAY, @mdate) = 1 AND DATEPART(MONTH, @mdate) = 11 THEN 1 – All Saints Day
WHEN DATEPART(DAY, @mdate) = 25 AND DATEPART(MONTH, @mdate) = 12 THEN 1 – Christmas Day 1
WHEN DATEPART(DAY, @mdate) = 26 AND DATEPART(MONTH, @mdate) = 12 THEN 1 –. Christmas Day 2
WHEN @mdatenum = @mostern - 2 THEN 1 – Good Friday
WHEN @mdatenum = @mostern THEN 1 – Easter Sunday
WHEN @mdatenum = @mostern + 1 THEN 1 – Easter Monday
WHEN @mdatenum = @mostern + 39 THEN 1 – Ascension
WHEN @mdatenum = @mostern + 49 THEN 1 – White Sunday
WHEN @mdatenum = @mostern + 50 THEN 1 – White Monday
WHEN @mdatenum = @mostern + 60 THEN 1 – Corpus Christi
ELSE 0
END
RETURN @mret
END

F. The SAP acceptable format of the date is DD.MM.YYYY (i.e. Day.Month.Year) and time are HH: MM (i.e. Hour: Minute) initially set during the processing phase of the data within the SQL Stored Procedure and further the style for the SAP excel template cells are updated using the server side SpreadsheetLight code.

//C# code snippet for formatting the date and time with respect to the final SAP booking template
4.4 Documentation

User documentation in the form of a user guide or an instruction manual finally designed for helping the end users to use the application system. The document includes written and other visual information about the application system, how it works and how to use it.

In this project the basic user manual (see Appendix 2) is written in the word document that mainly includes the following structure:

1. General Information:
   a. System Overview: This section details the purpose and scope of the application system.
   b. Point of Contact: This section specifies the organization staff in the help desk for telephone or email assistance for the system users.

2. System Summary:
   a. System Configuration: The fundamental development tools employed for the construction of the system is detailed here and also provide the URL for accessing the hosted web application Intranet portal within the company network.

3. Getting Started:
   a. Logging On: The URL link and the primary credentials (username and password) for pre-defined users are specified.
   b. System Operation: The main stages for generation of SAP booking template is explained using the screenshots from web application portal.
   c. System Validations: The validation messages or the internal check performed by the system detailed in this section.

4. Reporting:
a. Generated SAP Timesheet: The image of the final output SAP booking timesheet attached for user perception.

5 Results and Discussion

In this chapter, the results of the study presented and discussed in concurrence with the aim of the project, which was to enhance the existing timesheet data processing speed via stored procedure and to generate the SAP booking data file. The three main subsections, the first is to compare the time required to perform the distribution of the combined working hour of employees in the input file, to generate distributed working hour in the output SAP file by the stored procedure based on the fixed criteria as described in section 4.3.3. The second is to determine the average time required to generate the output file by the application system for a set of months. The third section is to examine the query execution time of the stored procedure on the SQL Server platform.

5.1 Employee Work Hour Distribution

The input timesheet file includes each record with the cumulative work hours of employees for each project they have operated in for the month. The pre-defined condition of cell splitting as explained in section 4.3.3 applied to each of these records by the stored procedure. The stored procedure and its associated user-defined functions perform the data processing of each row of data from the input file and generate a distributed time value in multiple rows in the resultant SAP booking output file.

Consider the cumulative work hour entry of an employee, say Employee_1 managed two projects in July 2018 (see Table 8).

Table 8: Input File Data

<table>
<thead>
<tr>
<th>Employee Name</th>
<th>Personal Number</th>
<th>Project Number</th>
<th>Cumulative Work Hour (CWH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee_1</td>
<td>580383</td>
<td>0821.2111</td>
<td>85:00</td>
</tr>
<tr>
<td>Employee_1</td>
<td>580383</td>
<td>0825.2511</td>
<td>30:00</td>
</tr>
</tbody>
</table>

The resultant SAP output after splitting the cumulative working hour based on the predefined condition shown in Table 9. The first record in Table 8 having the CWH of the employee is 85:00 for the Project Number 0821.2111 distributed into a set of eleven-row value representation in SAP booking file. In addition to it, the two weekend days eliminated in the result generated, here 07-07-2018 and 08-07-2018 are Saturday and Sunday respectively (see column Date in
Table 9). The timesheet file and the corresponding SAP booking file produced by the system are presented in Appendix 17 and Appendix 18 respectively.

Table 9: Output Booking Data

<table>
<thead>
<tr>
<th>#</th>
<th>Personal Number</th>
<th>Date</th>
<th>Begin (HH:MM)</th>
<th>End (HH:MM)</th>
<th>Duration (HH:MM)</th>
<th>Project Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>580383</td>
<td>02-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>2</td>
<td>580383</td>
<td>03-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>3</td>
<td>580383</td>
<td>04-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>4</td>
<td>580383</td>
<td>05-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>5</td>
<td>580383</td>
<td>06-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>6</td>
<td>580383</td>
<td>09-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>7</td>
<td>580383</td>
<td>10-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>8</td>
<td>580383</td>
<td>11-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>9</td>
<td>580383</td>
<td>12-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>10</td>
<td>580383</td>
<td>13-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0821.2111</td>
</tr>
<tr>
<td>11</td>
<td>580383</td>
<td>16-07-18</td>
<td>8:00</td>
<td>10:00</td>
<td>2:00</td>
<td>0821.2111</td>
</tr>
<tr>
<td>12</td>
<td>580383</td>
<td>16-07-18</td>
<td>10:00</td>
<td>16:18</td>
<td>6:18</td>
<td>0825.2511</td>
</tr>
<tr>
<td>13</td>
<td>580383</td>
<td>17-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0825.2511</td>
</tr>
<tr>
<td>14</td>
<td>580383</td>
<td>18-07-18</td>
<td>8:00</td>
<td>16:18</td>
<td>8:18</td>
<td>0825.2511</td>
</tr>
<tr>
<td>15</td>
<td>580383</td>
<td>19-07-18</td>
<td>8:00</td>
<td>15:06</td>
<td>7:06</td>
<td>0825.2511</td>
</tr>
</tbody>
</table>

It is quite clear that the number of rows generated in the output file for each record within the input file is dependent on the cumulative work hour value and observed that the number of rows created in SAP booking file increases with the rise in the CWH.

The CWH of a record in the input timesheet file is equivalent to the sum of the duration \((d)\) of the records in the total column split \((n)\) in SAP booking file.

\[
\text{Cumulative Work Hour (CWH)} = \sum_{k=1}^{n} d_k
\]

The scatter plot (see Figure 21) based on the data from the documents in Appendix 18 (having 232 input row value and 932-row value for its corresponding sap output file) shows that the cumulative work hour is directly proportional to the number of data record generated (since the slope in the graph is a straight line). The number of records/rows produced for the SAP booking file will always be higher than the number of records/rows for its corresponding input timesheet.
file.

![Employee Work Hour Distribution](image)

**Figure 21: Employee Work Hour Distribution**

5.2 **Average Execution Time**

The web application allows the users to generate the SAP booking data file mainly in two scenarios as mentioned in section 4.3.3. The user can either,

1. Upload the input timesheet file for a particular month and generate the respective SAP booking file.
2. Directly generate the SAP booking file by submitting the month and year for an already uploaded input timesheet file.

In the first scenario, the total execution time ($T_{total}$) is the sum of the time taken to save the input file data to the database ($T_{save}$) and the time needed to run the stored procedure ($T_{run}$).

$$T_{total} = T_{save} + T_{run}$$
Figure 22: Average Execution Time of Stored Procedure

In the second scenario, the total execution time \( T_{\text{total}} \) is equal to the time taken to run the stored procedure \( T_{\text{run}} \).

\[
T_{\text{total}} = T_{\text{run}}
\]

Figure 22 shows the execution for both scenarios for the uploaded timesheet file from March 2017 till July 2018. The inferred average execution time in the first scenario is nearly 3000 milliseconds (approximately 3 seconds) whereas for the second scenario the total time is directly proportional to the execution time of the stored procedure and therefore the net execution time of the stored procedure is approximately 1353 milliseconds or 1.353 seconds. In both the scenarios the values of time(T) provided depends on the execution time of both the server-side C# code and the stored procedure and its associated user-defined functions in the SQL Server.

5.3 SQL Server Execution Times

In SQL Server the consumed CPU time and Elapsed time associated with subsequent SQL queries run in the query window can be viewed in the message tab by executing the following command,

```sql
SET STATISTICS TIME ON;
```
The time taken for preparation of query through syntax check and creation of normalized task list is called Parse Time ($T_P$) and on the other hand, the time taken to run the query through algerbrizer is called Compile Time ($T_C$). Elapsed Time under SQL Server Execution Times refer to the total time taken to execute the query furthermore the CPU Time ($T_{CPU}$) is the sum of Parse Time and Compile Time.

$$T_{CPU} = T_P + T_C$$

Elapsed Time ($T_E$) is higher than CPU Time due to the waiting time consumed for data read from disk or memory or due to locking and blocking.

$$T_E > T_{CPU}$$

From Figure 23, it is clear that the Elapsed time will always be higher than the CPU time for the stored procedure. Here the CPU time lies between 280 ms and 550 ms whereas the Elapsed time lies between 780 ms and 970 ms. Therefore, the total execution time for the Stored Procedure on SQL Server interface nearly takes 1 second (i.e. equal to the maximum value Elapsed time).

![SQL Server Execution Time](image)

Figure 23: SQL Server Execution Time of Stored Procedure

---

4 This component performs name resolution and generates a query execution plan.


6 Conclusion, Evaluation and Recommendations

In this chapter, the closing comments on the proposed application system presented providing an overview and conclusion on the research methodology. The proposed system examined based on the existing system and in terms of its contribution to the field and its limitations. Recommendations made that could be explored in future research, mainly focusing on the integration of the web application system with the SAP Business ByDesign platform for faster data transfer.

6.1 Conclusion

The dependency on web applications is becoming so high that the people are switching from offline to online in every perspective of life since it serves the people by easing their tasks (Jailia et al. 2016, p.1). For many business web applications, relational database forms the primary repository with its robust data structure for retrieving, organizing and managing data (Yafooz et al. 2013, p.198). Relational databases handling a large amount of data, complex transaction rule and with high system security requirements use stored procedures to improve the efficiency, maintainability, data integrity and reliability (Tan Baohua and Zeng Ling 2010, p.595).

This thesis motivated by conducting the case study on the existing issues associated with the present timesheet management system based on Visual FoxPro at GITEC-IGIP GmbH. The consideration of an integrated, user-friendly online web application system that could accelerate the timesheet management process, lead to the concept of implementing an intranet portal for the company based on MVC.NET framework integrated with SQL Server stored procedure. However, the main aim of this project was to describe the role of the stored procedure for faster data processing and execution by reducing the network traffic.

The results show that the accuracy and data processing speed has increased with the implementation of the new web application portal using the SQL Server stored procedure in the Code First Approach of Entity Framework in MVC 5. The issues associated with the existing system adequately explained and designed a functional intranet web-based application portal for the employees of the company with the responsive interface and faster generation of the SAP booking file for recording the monthly employee work hour in SAP Business ByDesign Web portal.

6.2 Evaluation of the System

The main strengths of the application system are:
1. Minimal processing steps for generation of the final SAP booking file reduced the overall workload of the employees while compared with the old system. Users earlier had to memorize the steps and manually transfer the data in the intermediate processing stages. Now they have to upload the input data file and then the system internally executes all the processes and creates the final output SAP booking file.

2. The proposed system is built on Visual Studio having Microsoft support whereas the existing system in Visual FoxPro, an abandoned Microsoft programming environment. The web application developed with open source .NET web framework on Visual Studio platform and it supports the developers for faster development, maintenance and deployment of the web app.

3. Employees can use the web application on multiple devices like smartphones, desktop computers, iPad etc. due to its responsive design whereas the FoxPro based system is a desktop application with a poor user interface and had to be installed separately for its operation on the corresponding computer system.

4. The .NET web application framework along with SQL Server stored procedures to improve the overall processing speed of the system and generate data with high accuracy and precision.

5. SpreadsheetLight is an open source developer friendly open XML spreadsheet library for the .NET framework that performs server-side data formatting corresponding to the SAP template. In the old application system, the data formatting was one of the major problems confronted (refer section 3.1.3 Paragraph 4).

6. The users can generate the SAP booking template file for an already uploaded input timesheet file by providing the month and year details in the web page of the application. On the other hand, in the FoxPro system, the user had to repeat the same steps for generating the output SAP booking file.

The limitations of the system:

1. Lack of direct submission of the final booking data to SAP Business ByDesign web portal. Currently, the user after generation of the final SAP booking template upload the data using the Microsoft Excel interface to the SAP Business ByDesign web portal.
2. Creation of SQL stored procedure and user-defined functions with manual testing based on the test cases written manually and lack of proper code optimization. That might adversely affect the overall performance of the Relational Database Management System (i.e. MS SQL Server in our project).

3. Lack of support from traditional databases like MS SQL Server with an increase in the velocity of the data can affect the consistency of transactions.

4. To explore fully the query optimization based on stored procedures, further assessment on the stored procedure and its associated user-defined functions to generate optimized stored procedure query block. Consequently, further improves the performance of the Relational Database Management System (i.e. MS SQL Server in our project).

6.3 Recommendation for Further Research

Some of the aspects of further studies are:

1. The developers can further explore the support for the connectivity of SAP Business ByDesign web portal platform with the .NET framework for faster data transfer among the systems.

2. It would be interesting to make use of newer tools for automated execution of stored procedures that automate the generation of test cases and corresponding database states for the stored procedures (Mahmood et al. 2016, p. 519).

3. It would be interesting to assess the use of NoSQL (Not Only SQL) database within the system due to the lack of support from traditional database like SQL Server, used currently within the application for handling a large amount of data. In SQL databases the scalability is expensive due to greater hardware failure uncertainty and the cost of hardware in future upgradability resulting in high overall implementation cost with an increase in data growth whereas in case of NoSQL databases with the increment in the volume of data, the system expands by adding further nodes for data storage and processing power (Sahatqija et al. 2018, p.240).

4. The major drawback of the system is that its accessibility limited within the company network (being an intranet web portal). In future, the company GITEC-IGIP GmbH can host the web site on a paid web hosting server that can allow the users to access the web portal over the internet.
References


Wang, Y. and Zhang, X. (2012). The Research of Multi-source Heterogeneous Data Integration Based on LINQ. 2012 International Conference on Computer Science and Electronics Engineering, pp.147-150.


Declaration of Authenticity

I, Anith Shaji, hereby declare that the work presented herein is my own work completed without the use of any aids other than those listed. Any material from other sources or works done by others has been given due acknowledgement and listed in the reference section. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted. The work presented herein has not been published or submitted elsewhere for assessment in the same or a similar form. I will retain a copy of this assignment until after the Board of Examiners has published the results, which I will make available on request.