Software Development Plan

Attendance System

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Project Overview

The aim of this project is to develop a computer based attendance system capable of registering and verifying attendance to a wide range of event types. This could include students attending obligatory course events or workplaces holding compulsory meetings for employees. Often, such systems are implemented with attendances being manually recorded on paper. This approach can be both error prone and time consuming, especially when collating data and calculating individual levels of attendance for larger events.

The application is intended to reduce the time needed to record attendance and provide a more convenient way to create a report from the data collected.

1.2 Project Deliverables

- Checkpoint System
- Planning Documents
- UML Documents
- Presentation
- User Manual

1.3 Restrictions

- The system must be flexible enough to be used by anyone who needs an attendance system.
- The development of the system must be completed within the time frame of the spring semester (10th Jan – 16th May 2017).
- As a student project the budget is essentially negligible.
- The programming language must include C#

2 PROJECT ORGANIZATION

2.1 Team Description

Brueland, Kevin
mail: Kevin.brueland@gmail.com
tlf: 4128376
role: System Developer
Experience:

Liknes, Morten
mail: mliknes@hotmail.com
tlf: 47882901
role: System Developer
experience:

Lam, Chi Mon Noel
mail: tonje_@hotmail.co.uk
tlf: 99297412
role:
Experience:

Dæhli, Olav
mail: Olav.Dehli@usn.no
tlf: 35575182

Halvorsen, Hans Petter
mail: hans.p.halvorsen@usn.no
Tlf:35575158

2.2 Role Description

The team will be acting as both the client and the developer, as such each member of the team will be required to perform in various roles during the lifetime of the project. The course leaders, Hans P Halvorsen and Olav Dæhli, are the development management team.

- Client (Stake-Holder)
- Product Owner (voice of the client)
- Development Team
- Scrum Master
- Development Managment

2.3 Software Process Model Description

The project development will adhere to the Scrum strategy. This entails that the development process will be incremental and use a series of clearly defined iterations known as 'sprints' to deliver the product in phases. This has the benefit of making the development more manageable with clearly defined short term goals.

Once the initial requirements are established by the client they are added to the project backlog. At the start of every sprint the client has the opportunity to identify the system requirements

which are to be prioritized from the project backlog. Once these requirements are agreed upon they cannot be changed during the course of the sprint. At the end of each sprint the team is expected to deliver a fully functional version of the product which the client can then evaluate. This approach provides a level of flexibility that lets the client can make any necessary adjustments to the requirements for the next sprint. This process is repeated until the backlog is cleared.

A feedback driven strategy like Scrum helps all parties to avoid any confusion or unexpected surprises as is common with other development strategies such as the Waterfall method which is a sequential process wherein the client is largely only a participant in the initial planning stage and presented with a finished product at the end.

3 CODE AND NAMING CONVENTION

3.1 C#

The following coding and naming conventions apply to all code written in the C# language.

3.1.1 Variables

- All variables will be declared using camel case notation, i.e. myVariable.
- All private variables will be prefixed with an underscore.
- Variable names should be as descriptive as possible, avoiding unnecessary abstraction.

3.1.2 Methods

- All methods will be declared using Pascal Case/Upper Camel Case, i.e. MyMethod.
- Method names should be as descriptive as possible, avoiding unnecessary abstraction.
- Whenever possible, methods should be restricted to perform one task and with the minimum number of arguments, in line with the single responsibility principle. (No arguments being the preferred structure).

3.1.3 Classes

- All classes will be declared using Pascal Case/Upper Camel Case, i.e. Class.
- The contents of a class must be structured in the following order:

3.1.3.1 Fields

See *3.1.1 Variables*

3.1.3.2 Properties

- All properties should reflect the name of its associated field, and declared using Pascal Case/Upper Camel Case, i.e. MyProperty.
- Property names should be as descriptive as possible, avoiding unnecessary abstraction.

3.1.3.3 Events

- All events will be declared using Pascal Case and the associated event handler will be declared using the same name, but being prefixed with the word "On" and in past tense, i.e. OnButtonClicked.

3.1.3.4 Methods

See 3.1.2 Methods

3.1.4 Interfaces

- All interfaces will be prefixed with the letter "I".
- All interfaces will be declared using Pascal Case/Upper Camel Case, i.e. *IMyInterface.*
- Interface names should be as descriptive as possible, avoiding unnecessary abstraction.
- All interfaces must contain the name of the classes which will implement the interface.

3.1.5 Enumerations

- All enumerations will be declared using all lower case letters, i.e. myenumeration.

3.2 SQL

The following coding and naming conventions apply to all code written in SQL.

3.2.1 Tables

- All table titles will be capitalized, i.e. TABLE.
- All table titles will be nouns of a singular form.

3.2.2 Attributes

- All attribute names will be in Pascal Case/Upper Camel Case, i.e. MyAttribute.

3.2.3 Stored Procedures

- All stored procedures will be in Pascal Case/Upper Camel Case and their associated variables in camel case.

4 RISK ANALYSIS

In order to maximize development work flow and avoid unnecessary project delay, a risk analysis was performed and can be viewed in *Table 1: Possible risk factors and preventative measures*

Table 1: Possible risk factors and preventative measures

Risk Factor	Preventative Measure
Unrealistic schedule	Business-case analysis, incremental development
Overestimation of IT skills	Technical analysis, prototyping
Possible system conflicts when using off-the-shelf products (drivers)	Compatibility analysis , alternative product choices
Unforeseen changes in HR (illness, relocation)	Good communication, remote participation possibilities
Hardware malfunction	Centralized backup

5 SOFTWARE AND HARDWARE RESOURCE REQUIREMENTS

This chapter gives a general overview of the software and hardware required to develop the software. A list of specific items based on the general list in this chapter, can be found in *10 Tools.*

5.1 Software

- Programming environment that supports C#.
- Database modelling program.
- Database management program.
- Documentation & word processing software.
- Shared file hosting & collaboration service.

5.2 Hardware

- Unique identification system.
- Computer running Windows OS.

6 WORK BREAKDOWN

The development of the CheckPoint Attendance Software is divided into «milestones», as described in *Figure 1:Work Breakdown Structure*

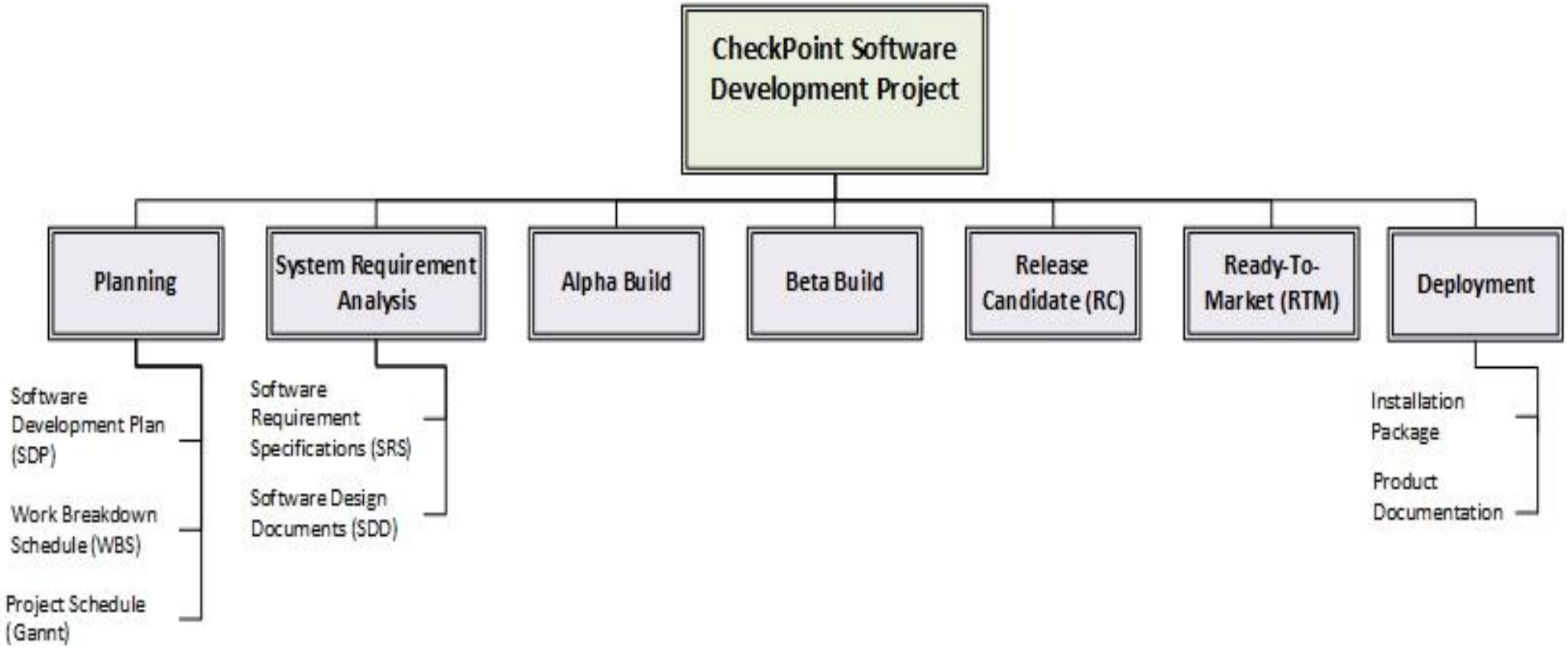

Figure 1:Work Breakdown Structure

7 PROJECT SCHEDULE

The Gantt chart or the project schedule chart is a dynamic document and is subject to change during the entirety of the development period. The lastest version of the project schedule can be found in *Figure 2: Project schedule plan*

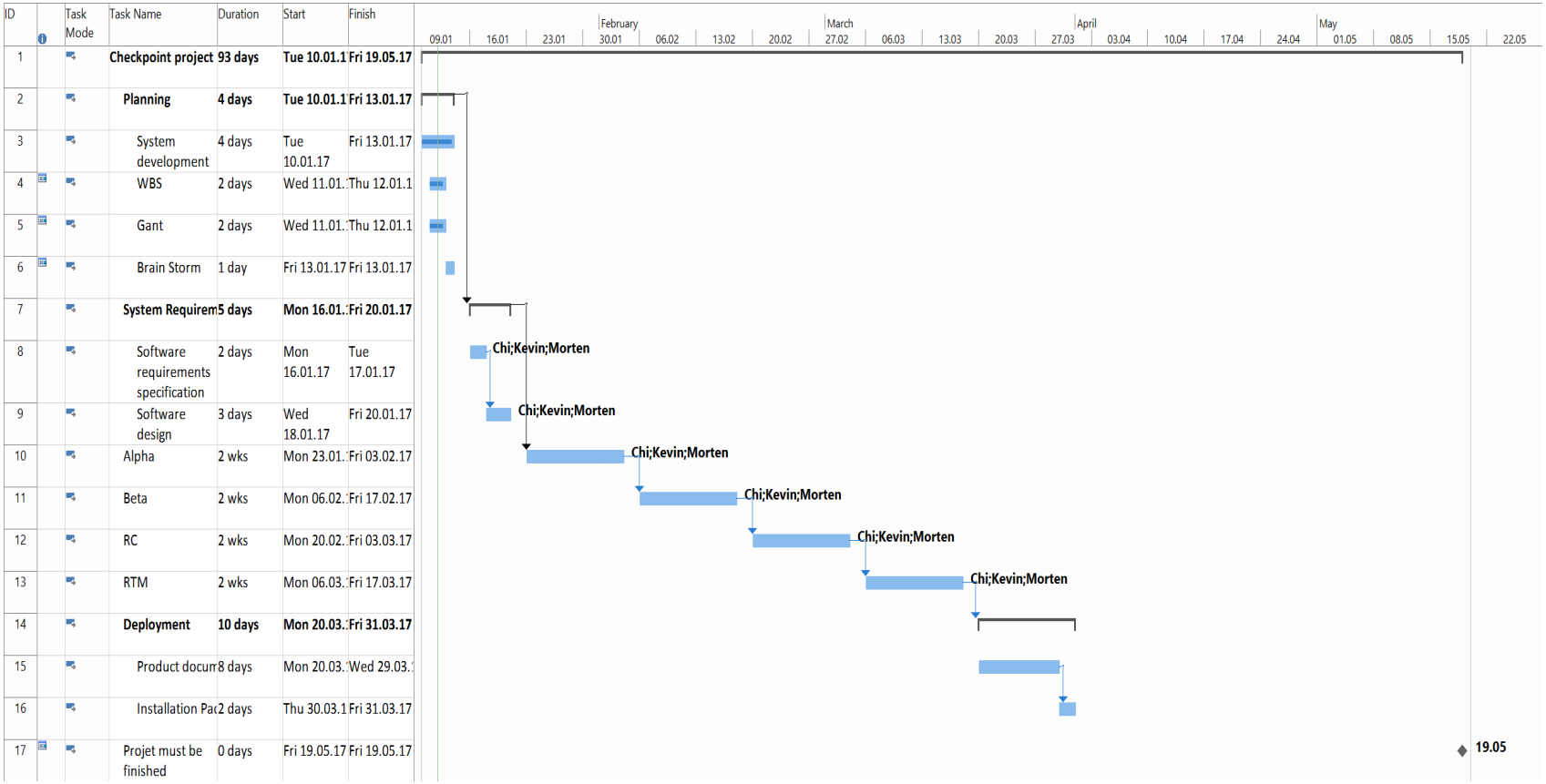

Figure 2: Project schedule plan

8 MONITORING AND REPORT MECHANISMS

Visual Studio Team Server (VSTS) will serve as the common development platform for monitoring project progress and bug reporting. It will provide a centralized storage location for hosting documents and managing code revisions.

Address for Team Server:

http://www.checkpointas.visualstudio.com

In addition, the development team has fixed office hours every Tuesday and Friday at 10.15 – 14.00.

All documentation related to the software development shall adhere to the document template "CheckPoint Attendance System Template" word document located in the VSTS document folder.

9 TESTING

The aim is for the entire code to be unit testable. This requires comprehensive decoupling of dependencies system-wide. Due to the restrictions mentioned earlier in the SDP, it is unrealistic to expect the team to perform complete unit tests of the software, but the team will endeavor to carry out a reasonable level of testing, focusing on the primary features.

Both functionality and user experience (UX) testing will be carried out to ensure a level of convenience is maintained.

The majority testing will be conducted within the individual iterations/sprints.

10 TOOLS

This section contains the required software and hardware needed to develop and complete the targeted software.

10.1 Software

It is expected that all project members have the software listed in *Table 2: Required software*

Table 2: Required software

Objective	Software	Version
IDE	**Visual Studio Enterprise Edition**	**2015 with update 3**
IDE	**Arduino Sketch**	**1.6.7**
Database Modelling Tool	**CA Erwin Data Modeller**	**r9.64**
Database Management System	**Microsoft SQL Server Management Studio**	**2014**
Documentation	**Microsoft Office 365**	**N/A**
General communication	**Facebook Messenger**	**N/A**
Voice/video communication	**Microsoft Skype for Business**	**2016**
Document collaboration	**Microsoft OneDrive**	**N/A**

10.2 Hardware

The hardware listed in *Table 3: Required harware* is used in the development project and forms the basis for all aspects of compatibility and testing.

Table 3: Required harware

Hardware	Version/Specifications
Intel NUC	**tba**
RFID reading unit	**RC522 Module**
RFID tags	**Passive,13.56MHz**
Arduino Uno	**Rev 3**
Windows-Based Computer	**Installed with Microsoft Windows 7 or newer**