**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 07.10.2015 | 1.0 | Software Development Plan (Version 1.0) | Caner Gülgeç  Arjen Kılıç  Cengizhan Özcan  Erke Varol  Şevval Tan |
| 11.10.2015 | 2.0 | Software Development Plan (Version 2.0) | Caner Gülgeç (Identification and Software Development Process)  Arjen Kılıç (Software Design, Software Development Rules & Standards)  Cengizhan Özcan (Responsibilities)  Erke Varol (Risk Assessment)  Şevval Tan (General overlook on the report and checking the requirements)  Everyone -> Software Development Tools |

**TABLE OF CONTENTS**

**Revision History**

**1** **Identification**

***1.1*** ***Document overview***

***1.2*** ***Abbreviations***

1.2.1 Abbreviations

***1.3*** ***References***

1.3.1 Project References

**2** **Software Development Activities**

***2.1*** ***Software development process***

2.1.1 Overview of process phases

2.1.2 Technical documentation

2.1.3 Deliverables

***2.2*** ***Software development tools***

2.2.1 Workstation

2.2.2 Requirements management and documentation

2.2.3 Software Design

2.2.4 Coding and automated tests

2.2.5 Configuration management

***2.3*** ***Software development rules and standards***

**3** **Responsibilities**

***3.1*** ***Activities and responsibilities***

**4** **Risk Assessment**

***4.1*** ***Risk Analysis***

***4.2*** ***Risk Planning***

1 **Identification**

* 1. ***Document overview***

This document contains the software development plan of Exam & Homework Information Tracking System. It is a desktop application presenting the dates of the assignments and exams of the courses which is chosen by students for the semester.

* 1. ***Abbreviations***
     1. **Abbreviations**

SDP : Software Development Plan

SRS : Software Requirements Document

STP : Software Test Plan

SDD : Software Design Document

STR : Software Test Report

JSON: JavaScript Object Notation

UML: Unified Modeling Language

EHITS: Exam & Homework Information Tracking System

* 1. ***References***
     1. **Project References**

|  |  |  |
| --- | --- | --- |
| # | Document Identifier | Document Title |
| [R1] | 1 | WindowBuilder - https://eclipse.org/windowbuilder/ |
| [R2] | 2 | Ozyegin SIS - https://sis.ozyegin.edu.tr |
| [R3] | 3 | Ozyegin LMS - https://lms.ozyegin.edu.tr |

1. **Software Development Activities**

The section lists and describes the software development activities of Exam & Homework Information Tracking System software.

The software development process chosen for this project is the waterfall programming model.

## 2.1 Software development process

The waterfall programming model was chosen for the reasons below:

* Using the waterfall model will be more efficient in order to keep up with the course schedule.
* The requirements for the project are not tentative, therefore changes will not be extremely necessary for the project.

* + 1. **Overview of process phases**

The lifecycle of the EHITS project is composed of

* Purchasing the server to hold the JSON data.
* Parsing the JSON data which keeps the information about the course code, course name, exam and homework dates, etc.
* GUI implementation of the project.
* EHITS implementation (coding).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Task | Effort (by person) | Duration (days) | Dependencies |
| 1 | Purchasing the server | Everyone (all members will contribute to buy the server) | 1 | - |
| 2 | Parsing the JSON data | Caner Gülgeç, Arjen Kılıç | 2 | 1 |
| 3 | Performing unit tests | Erke Varol, Cengizhan Özcan | 7 | - |
| 4 | GUI implementation | Şevval Tan, Caner Gülgeç | 7 | - |
| 5 | EHITS implementation | Cengizhan Özcan, Arjen Kılıç, Erke Varol | 7 | 1,2,3,4 |
| 6 | Testing the software | Everyone | 4 | 1,2,3,4,5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Starting Date | Due Date | Duration (days) |
| SDP | 09/30/15 | 10/07/15 | 7 |
| SDP (v2) | 10/07/15 | 10/14/15 | 7 |
| SRS + STP | 10/14/15 | 10/21/15 | 7 |
| SRS + STP (v2) | 10/21/15 | 10/28/15 | 7 |
| SDD | 11/04/15 | 11/11/15 | 7 |
| SDD (v2) | 11/11/15 | 11/18/15 | 7 |
| STR | 12/09/15 | 12/16/15 | 7 |
| Code implementation | 11/18/15 | 12/16/15 | 28 |

* + 1. **Technical documentation**

The following documentation is produced during the design phases:

* Software specification: SDP, SRS, STP
* Software detailed conception: SDD, updated SRS and updated STP
* Coding and unit tests: STR of unit tests
* Software tests phrases : STR
  + 1. **Deliverables**

The following items are delivered at the end of the process:

* Technical documentations (SDP, SRS, STP, SDD, STR)
* Software and its configuration files.
  1. ***Software development tools***
     1. **Workstation**
* Caner Gülgeç: Lenovo, Intel Core i7, 2.4Ghz, 8GB RAM, Windows 10
* Erke Varol: Fujitsu, Intel Core i7, 2.5Ghz, 8GB RAM, Windows 8
* Şevval Tan: Apple, Intel Core i7, 2.5Ghz, 16GB RAM, OS X El Capitan
* Arjen Kılıç: Monster, Abra a5 v1.1 i7, 2.5Ghz, 8GB RAM
* Cengizhan Özcan: Apple, Intel Core i7, 2.2Ghz, 16GB RAM, OS X Yosemite

2.2.2 **Requirements management and documentation**

* Microsoft Word: To present the necessary documents.
* GitHub: To update and share the project with the team members.

2.2.3 **Software Design**

* ArgoUML will be used to present a UML diagram.

2.2.4 **Coding and automated tests**

Following tools will be used for coding and automated tests:

* Eclipse
* XCode: for Mac users

2.2.5 **Configuration management**

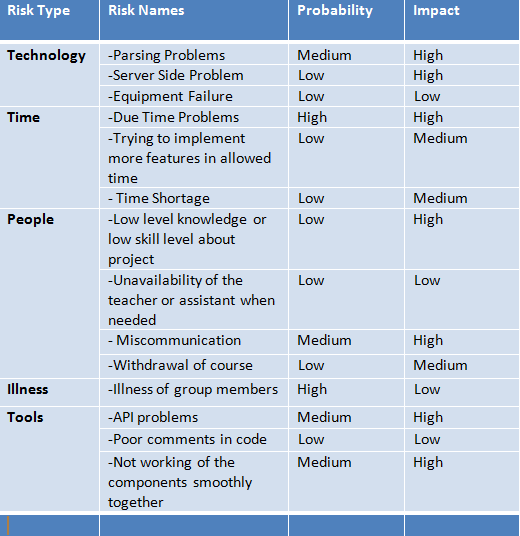
* GitHub: We will use a common repository location (on a desktop application) to keep track of the changes made on the software.
* Organization name: OzyeginCS320
* Repository name: ExamHomeworkTracker
* Google Docs: The platform which holds the updated versions of the reports.
  1. ***Software development rules and standards***
* CamelCase will be used as a coding standard.
* JSON will be used as a data model.
* UML diagram will be used for modeling the project.

1. **Responsibilities**
   1. ***Activities and responsibilities***

Below is a table that represents member activities.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Responsible Member/s** | **Comment** |
| Project management | Arjen Kılıç |  |
| Configuration tools management | Erke Varol |  |
| Setting up the Development tools | Erke Varol, Cengizhan Özcan |  |
| Software specifications | Arjen Kılıç, Cengizhan Özcan |  |
| GUI Implementation | Şevval Tan, Caner Gülgeç |  |
| Parsing the JSON data | Caner Gülgeç, Arjen Kılıç |  |

1. **Risk Assessment**
   1. ***Risk Analysis***



* 1. ***Risk Planning***
* **Parsing Problems**
* **Explanation:** Project will be based on parsing some data from Internet which is a new area for members.
* **Prevention:** Working on parsing subject and looking into some parsing technics or looking into some parsing API’s which can help in the project.
* **Correction:** When some problems or error starts to occur in the parsing, the project should be halted until group members can get some help from an expert.

* **Server Side Problems**
* **Explanation:** In project we will use server to save some data. Server based problems may occur. Server can be crashed.
* **Prevention:** All data will also be stored in the project repository, which is backed up regularly by the leader.
* **Correction:** When the data loss happens it can be recovered by obtaining the most recent backup.

* **Equipment Failure**
* **Explanation:** In project, the equipment used by the members can be broken or there may happen some failures.
* **Prevention:** Each group member should do their regular updates, anti-virus scans, backups and their hardware check.
* **Correction:** Members can continue working on another computer with their backups.

* **Due Date Problems**
* **Explanation:** Due dates are strict but group members may miss the deadlines according to their heavy schedule.
* **Prevention:** All group members should put some reminder on due dates and group leader should remind the deadline regularly. If a group member thinks he or she going to be unable to do the job until the deadline, that member should tell this to his or her leader.
* **Correction:** If this case happens group leader should rearrange the assignments until the due date and warn that group members.

* **Trying to implement more features in the allowed time**
* **Explanation:** Members can get ambitious and try to implement some more features into the program in the allowed time and resource constraints.
* **Prevention:** Group members should try to implement required features firstly. Tasks should be prioritized. Maybe group can leave some time for additional features.
* **Correction:** At worst case scenario we will cut some features with low priority.

* **Time Shortage**
* **Explanation:** Group can underestimate the time which is needed to complete the software.
* **Prevention:** Group should plan all the assignments like writing reports and coding process before and create a time schedule for this project.
* **Correction:** If the time begins to be a problem for the group some low priority requirements can be dropped like in the previous risk.

* **Low level knowledge or low skill level about project**
* **Explanation:** Group members may not have enough knowledge or skills about the project subject.
* **Prevention:** Group leader or most experienced person in the group should give some detailed information about the project before the project starts so members can work on their weaknesses.
* **Correction:** Group can change their approach to the project or take some help from an expert.

* **Unavailability of the teacher or assistant when needed**
* **Explanation:** Teacher or assistant may be unavailable at the required time.
* **Prevention:** Meetings with the teacher or assistant should be arranged by talking or through e-mail before some time, not in that day.
* **Correction:** Different appointments can be arranged again or group may ask help from another teacher or assistant at that time.

* **Miscommunication**
* **Explanation:** Misunderstanding or miscommunication can occur in the group after meetings or some disagreements while or after the meetings. Group members can gossip about each other when the things don’t go correctly.
* **Prevention:** While at meeting one group member can take notes of the discussed subjects and at the end he can copy them and give it to other members so that all the members can be on the same track with each other. Also according to ethical rules gossip shouldn’t even be an issue.
* **Correction:** When group realizes that miscommunication starts to cause problems across the group, group should immediately arrange a meeting and talk about the problems in their heads. All the things should be clear at the end.

* **Withdrawal of course**
* **Explanation:**  Some of the team members can drop the course.
* **Prevention:** Working with solid group members and trusting each other and to the project at the start should prevent this risk. Also if the project is well understood there will be no one in the group who is going to drop the course.
* **Correction:** Leader should distribute the tasks among the group members again. If all the group members are well aware of each others tasks already this will not be much of a problem. Also with a smaller group, group can be managed easier and extra work should be handled easier.

* **Illness or absence of group members**
* **Explanation:** Some group members can get sick or can be absence at the meeting according to some reasons.
* **Prevention:** If a member misses a meeting because of illness they should try to inform the group leader as soon as possible.
* **Correction:** If the knowledge in the group isn’t a problem then group leader can assign works again and work can be taken over by someone else very fast. Planning is the key in this.
* **API Problems**
* **Explanation:** Group members can have some problem with APIs. They may not have enough knowledge about them or how to use them in exact situations.
* **Prevention:** Group members should be well prepared for the APIs which are going to be used in the project and be familiar with them. Some practice might be useful.
* **Correction:** Group can ask for help from an expert or some member who is confident with that API and who could handle APIs in the project and also work distribution should be done once again.

* **Poor comments in Code**
* **Explanation:** Comments in the code may be inefficient to understand the code.
* **Prevention:** Comments should be open and clear for the instructor and the assistant to understand what is going on in the code.
* **Correction:** If the instructor didn’t understand a point in the code even with comments he or she can directly talk with the group leader.
* **Not working of the components smoothly together**
* **Explanation:** Some components may not work smoothly together. For example some problems about the APIs may occur with the code which is going to implemented by the group members.
* **Prevention:** Planning of the implementation of code pieces and APIs should be done earlier and group should consider if there are going to be any problems. Components should also be tested before.
* **Correction:** Code should be written again and components should be checked. Maybe the group should search for some other APIs to work with.