|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | | | | |
|  | |  | | | |
| Project: | | Personal Health Monitoring System  (PHMS)  CSE 5325 – Spring 2023  Project Management | | | |
| Module: | | COCOMO | | | |
| Deliverable: | | COCOMO Estimate Report | | | |
| Version: | | | [1.0] | Date: | [04/11/2023] |

Prepared by: Sai Parthish Mandumula and 1002022847

TABLE OF CONTENTS

[1. Introduction 2](#_Toc23340379)

[2. Estimating Factors 3](#_Toc23340380)

[2.1 Source of Lines of Code 3](#_Toc23340381)

[2.2 Scale Drivers 3](#_Toc23340382)

[2.3 Cost Drivers 3](#_Toc23340383)

[3 Project Final Timeline and Cost Structure 4](#_Toc23340384)

[4. Conclusion and Recommendations 5](#_Toc23340385)

[Appendices 6](#_Toc23340386)

# 1. Introduction

The main aim of this document is to estimate the **Personal Health Monitoring system** cost, effort, and timeline (SMS).

Gathering information on the project's needs, the size and complexity of the project, the expertise of the development team, and the development tools will allow you to estimate the cost, effort, and timeframe of the Personal Health Monitoring System. The COCOMO (COnstructive COst MOdel) model is widely used for calculating the time and money needed to complete a software development project. To calculate how much time and money will be needed to complete a project, COCOMO considers several variables. These include the project's scope and complexity, the skill level of the development team, and the tools available to them. These are just rough estimations and might change as the project develops. As a result, it is essential to keep an eye on the estimates and make adjustments as needed during the project's lifetime.

**To get COCOMO findings, the following three crucial factors were used:**

* Source line of code (SLOC)
* Scale driver
* Cost drivers

2.1, 2.2, and 2.3 provide a thorough explanation of these three factors.

**Estimates produced by COCOMO that are effective:**

The sum of the following factors:

* The total size of the SLOC
* The total duration
* The total effort
* The total productivity in the developed SLOC per person-month
* The overall cost

**Model and methodology employed by COCOMO**

In order to complete this project, we employed the waterfall methodology.

For this project, we applied the SLOC methodology.

Five (5) developers were employed by us.

# **2. Estimating Factors**

## **2.1 Source of Lines of Code**

The following is the number of lines of code delivered as part of this project, A justification for the total amount of LOC is provided.

|  |  |
| --- | --- |
| **SLOC | Source Lines of Code** | Value Chosen: 5000 |
| Justification: A software project's size may be measured in terms of source lines of code (SLOC). Lines of code (including comments and blank lines) are tallied to determine the total size of the program. A software system's SLOC may stand in for the time and money needed for its development, testing, and upkeep. Notwithstanding these caveats, lines of code (or SLOC) continue to be a popular statistic in the software development world. It may aid project managers in making plans and allocating resources by providing a ballpark figure for the scope of the project. To acquire a more precise estimate of the time and money needed to create and maintain a software system, it should be used in combination with other metrics and factors. | |

## **2.2 Scale Drivers**

The following is the list of scale drivers, the values applicable to this project, and a justification for each value chosen:

|  |  |
| --- | --- |
| **PREC | Precendentedness** | Value Chosen: Generally Familiar |
| Justification: The COCOMO (COnstructive COst MOdel) relies on a collection of elements known as "Scale Drivers" to calculate an approximation of the time and money needed to create a piece of software. Priority, Development Agility, Architecture/Risk Resolution, Team Unity, Process Maturity, and Needed Reusability are the six Scale Drivers. | |

|  |  |
| --- | --- |
| **FLEX | Development Flexibility** | Value Chosen: Some Relaxation |
| Justification: The COCOMO (COnstructive COst MOdel) is a model used to predict the time and money needed to build software. It is a measure of how much room for maneuver is needed as the project evolves. Projects with limited Development Flexibility have clear needs and few potential areas for modification. Less work and money are expected to be spent on these projects since their development processes are more simple and there are less risks and uncertainties. The time and money needed to manage changes and adapt to the project's developing demands might increase the more Development Flexibility a project has. Planning a software project correctly involves taking Development Flexibility into account to avoid going over budget or behind schedule due to unforeseen changes. | |

|  |  |
| --- | --- |
| **RESL | Architecture/Risk Resolution** | Value Chosen: Often (60 %) |
| Justification: More time and money may be needed to manage project risks and address architectural problems in projects with a high Architecture/Risk Resolution. Consequently, it is vital to examine the degree of Architecture/Risk Resolution when planning a software project to ensure that sufficient resources are given to mitigate project risks and limit the likelihood of cost and schedule overruns. | |

|  |  |
| --- | --- |
| **TEAM | Team Cohesion** | Value Chosen: Basically cooperative |
| Justification: The COCOMO (COnstructive COst MOdel) is a tool for estimating the time and money needed to create software by factoring in the level of team cohesion present during development. It's a measurement of how well everyone in the development team can talk to one another and work together. If the team has poor cohesion, then it may take longer and more resources to plan for and complete the project. This is because of the increased likelihood of disagreements and misunderstandings. Consequently, it is essential to take Team Cohesion into account during software project planning to make sure the team is cohesive, works well together, and can accomplish its objectives in a timely manner. | |

|  |  |
| --- | --- |
| **PMAT | Process Maturity** | Value Chosen: SEI CMM Level 2 |
| Justification:  Estimating the time and money needed for a project with low Process Maturity might be difficult since it may take more time and money to set up a development process and guarantee adherence to standards and procedures. Planning a software project requires careful consideration of the Process Maturity level to guarantee a well-defined, mature, and capable development process that will efficiently and effectively support the project's objectives. | |

## **2.3 Cost Drivers**

The following is the list of cost drivers, the values applicable to this project and a justification for each value chosen:

|  |  |
| --- | --- |
| **ACAP | Analyst Capability** | Value Chosen: Very High |
| Justification: More time and money may be needed to define requirements and manage uncertainties before an accurate effort and cost estimate can be made for a project with poor Analyst Capabilities. Consequently, it is essential to take Analyst Capacity into account during software project planning to guarantee that the system analysts are adequately educated, experienced, and skilled to specify the project's needs. | |

|  |  |
| --- | --- |
| **APEX | Applications Experience Cost Driver** | Value Chosen: Very High |
| Justification: Estimating the time and money needed to complete a project when Application Experience is low may need extra time and money spent on training and education. Hence, the amount of Application Experience should be taken into account during software project planning to guarantee that the development team has the necessary experience and knowledge to create the application efficiently and effectively. | |

|  |  |
| --- | --- |
| **PCAP | COCOMO Programmer Capability Cost Driver** | Value Chosen: Very High |
| Justification: Cost and time estimates may need more time and money if the development team has to be better equipped or trained in order to complete a project with a low Capabilities Cost Driver. Hence, the Capacity Cost Driver must be considered throughout the software project planning phase to guarantee that the development team has access to the necessary tools, resources, and development environments. | |

|  |  |
| --- | --- |
| **PLEX | Platform Experience Cost Driver** | Value Chosen: Very High |
| Justification: The team comprises professional experts in AWS, UX/UI path, embedded systems, IoT, front end, and backend. | |

|  |  |
| --- | --- |
| **LTEX | Language and Tool Experience Cost Driver** | Value Chosen: Very High |
| Justification: Our developers are knowledgeable in React, larval, JavaScript, Java, React, Springboot, MySQL, HTML, CSS, and other programming languages. | |

|  |  |
| --- | --- |
| **PCON | Personnel Continuity Cost Driver** | Value Chosen: Very High |
| Justification: We anticipate bringing on several innovative employees in the near future. We're considering hiring testers because our Staff Software Developers will be doing double roles and responsibilities as testers for the first release. As a result, our chosen value is very high. | |

**Platform:**

|  |  |
| --- | --- |
| **TIME | Execution Time Constraint Cost Driver** | Value Chosen: Extra high |
| Justification: We strive to create software programs that utilize as little available execution time as possible. As a result, TIME has been set to nominal. However, it also means that execution must not be subpar, or we will not be able to achieve customer expectations. | |

|  |  |
| --- | --- |
| **STOR | Main Storage Constraint Cost Driver** | Value Chosen: High |
| Justification: Especially since this is our first release, we don't want our software to take up a lot of space and want it to use half of the primary storage space. | |

|  |  |
| --- | --- |
| **PVOL | Platform Volatility Cost Driver** | Value Chosen: Low |
| Justification: We have chosen the Platform volatility cost driver(PVOL) to be low because there is not going to be any major change for a year after the project is deployed. we will focus on minor changes such as cosmetic changes, new versions, license renewal, performance scalability etc. every month. | |

**Product:**

|  |  |
| --- | --- |
| **RELY | Required Software Reliability Driver** | Value Chosen: Nominal |
| Justification: The RSR Driver is typically defined as the level of software dependability required to satisfy the requirements of the consumer or end-user. This driver is used to determine the appropriate level of resources that must be allocated to ensure that the intended level of software reliability is attained. To determine the RSR Driver, project managers must consider factors such as the consequences of software failure, the cost of software failure, and user expectations for reliability. | |

|  |  |
| --- | --- |
| **DATA | Database Size Cost Driver** | Value Chosen: Very High |
| Justification: As the scale of the database impacts the complexity of software design, coding, testing, and maintenance, it can have a significant effect on the total cost of the project. The Database Size Cost Driver considers factors such as the number of tables, the number of records in each table, and the complexity of the inter-table relationships. The Database Size Cost Driver is a crucial aspect of project planning because it can have a substantial effect on the overall budget and schedule. When estimating the development and maintenance costs of a project, project managers should carefully consider the database's size and complexity. | |

|  |  |
| --- | --- |
| **CPLX | Product Complexity Cost Driver** | Value Chosen: low |
| Justification: In project planning, the Product Complexity Cost Driver quantifies the impact of a software product's complexity on its development and maintenance costs. The COCOMO (Constructive Cost Model) estimation methodology uses it as one of the cost determinants. The Product Complexity Cost Driver is a crucial factor in project planning, as it can have a substantial effect on the project's budget and timeline. When estimating the project's development and maintenance costs, project managers should consider the software product's complexity. | |

|  |  |
| --- | --- |
| **RUSE | Required Reusability Cost Driver** | Value Chosen: Extra High |
| Justification: The Required Reusability Cost Parameter quantifies the impact of a software project's required level of reusability on its development and maintenance costs. It is one of the COCOMO (Constructive Cost Model) estimation model's cost determinants. The Required Reusability Cost Component is an essential factor in project planning, as it can have a significant impact on the overall budget and schedule. When estimating the project's development and maintenance costs, project managers should carefully consider the required level of reusability. | |

|  |  |
| --- | --- |
| **DOCU | COCOMO Documentation Match to Life-Cycle Needs Cost Driver** | Value Chosen: Very Low |
| Justification: First and primarily, because RUSE is high, DOCU must be nominal. Furthermore, in our Project Scope and Feasibility, we previously said that we would not be focused on a high-level system guide. This is a phase in which we wish to spend a reasonable period. | |

**Project:**

|  |  |
| --- | --- |
| **TOOL | COCOMO Use of Software Tools Cost Driver** | Value Chosen: Very High |
| Justification: The Required Reusability Cost Parameter quantifies the effect of a software project's reusability requirements on its development and maintenance costs. It is a factor in the COCOMO (Constructive Cost Model) cost estimation model. The Required Reusability Cost Component is a crucial aspect of project planning because it can have a substantial impact on the overall budget and schedule. When estimating the project's development and maintenance costs, project managers should take the required level of reusability into account. | |

|  |  |
| --- | --- |
| **SITE | COCOMO Multisite Development Cost Driver** | Value Chosen: Very High |
| Justification: The COCOMO Multisite Development Cost Driver is a crucial factor in project planning, as it can have a substantial effect on the total project cost and schedule. When estimating the project's development and maintenance costs, project managers should carefully consider the complexities of multisite development and devise strategies to manage the resulting complexity and communication issues. | |

|  |  |
| --- | --- |
| **SCED | Required Development Schedule Cost Driver** | Value Chosen: Low |
| Justification: The Required Development Schedule Cost Driver reflects the impact of schedule constraints on the development and maintenance expenses of a software project. It is one of the COCOMO (Constructive Cost Model) estimation model's cost determinants. The Required Development Schedule Cost Driver is an essential factor in project planning, as it can have a significant impact on the overall budget and schedule. When estimating, project managers must closely consider schedule constraints. | |

# **3 Project Final Timeline and Cost Structure**

- -Previous Cost, Work and Duration (from assignment #2).

|  |  |
| --- | --- |
| Duration | 3 months |
| Human Resource Cost | $67,450 |
| Non-Human Resources Cost | $15,700 |
| Profit (0.5\*(human+non-human cost)) | $83,150\*0.5 = $41,575 |
| Total cost | $83,150 + $41,575 = $124,725 |

- **New Time and Cost Structure**

|  |  |
| --- | --- |
| New Schedule (Duration) | 6 months |
| COCOMO estimated costs (Human Resources)  (requirement+product design+Detailed design + code/unit test+integration testing) | $86,600.00 |
| Non-Human Resources | $15,700 |
| Maintenance Cost | $5500 |
| Profit (0.5\*(human+non-human cost)) | $107,800\*0.5 = $53,900 |
| Total cost | $107,800+ $53,900 =  $161,700 |

# **4. Conclusion and Recommendations**

## **CONCLUSION: -**

In our Project Scope and Feasibility Analysis, a budget of $83,150.00 was established. We exceeded our budget, however, when we estimated manually. This is due to the fact that we did not initially estimate project costs and timelines based on the following factors:

• personnel

• project

• platform

• product.

This project contains Human resources allocated for a three-month project, and in the previous version of this project, no formulas were used. We underestimated the cost of non-human resources such as hardware, utilities, and licenses. We omitted the cost of maintenance. We were unaware of the productivity of the involved resources.

The following are the results of COCOMO:

• Total SLOC size: 5000

• Duration total (in months):6

• Total Cost (thousand dollars):86,6

**Recommendation:**

The following are the reasons we would employ COCOMO:

• Accurate forecast project total expense.

• Appropriate for our prototype project with a strict deadline

• Steps and specifics are simple to follow.

• Reports and graphs are generated easily and automatically, such as activity reports, etc.

• Individual team members and hardware and software are considered.

Despite the benefits of COCOMO, we will SPLIT the project between the original estimate and the COCOMO estimates.

In COCOMO's calculations, the cost of software licensing, facility rent, and other expenses was not included. However, we must be parsimonious, as these services will be quite expensive. An equitable and practical COCOMO-based initiative, for instance, cannot exceed one year.

# Appendices

Additional documents, screen shots of COCOMO.

Table

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Graphical user interface, table

Description automatically generated

Graphical user interface, table

Description automatically generated

Table

Description automatically generated