**Software Project Plan**

**Michigan Treasury Local Government Data Parser**

**Senior Design I - Winter 2021**

**Table of Contents**

[**1.0 Introduction**](#_heading=h.q3imkqtuybh) **3**

[1.1 Project scope](#_heading=h.rgefimjhxwji) 3

[1.2 Major software functions](#_heading=h.xw4ng9ph1phe) 3

[1.3 Performance/Behavior issues](#_heading=h.1gg0of4y6k9w) 3

[1.4 Management and technical constraints](#_heading=h.3tvaliddxph2) 3

[**2.0 Project Estimates**](#_heading=h.eh4wr5r52c1x) **4**

[2.1 Historical data used for estimates](#_heading=h.80tnecoej5o4) 4

[2.2 Estimation techniques applied and results](#_heading=h.nkhamxkjk0c9) 4

[2.2.1 Estimation technique m](#_heading=h.umxnlfph7q52) 4

[2.2.2 Estimate for technique m](#_heading=h.hywtod59j16x) 4

[2.3 Reconciled Estimate](#_heading=h.orl5bk4zysa4) 4

[2.4 Project Resources](#_heading=h.4kzuz5o6k1ac) 4

[**3.0 Risk Management**](#_heading=h.g6gli5xqid02) **4**

[3.1 Project Risks](#_heading=h.66wtxp2ehw9h) 4

[3.2 Risk Table](#_heading=h.xcjeujwnf7g5) 4

[3.3 Overview of Risk Mitigation, Monitoring, Management](#_heading=h.qd1g8z93ujt4) 5

[**4.0 Project Schedule**](#_heading=h.885luqti3971) **5**

[4.1 Project task set](#_heading=h.5c7godpv5dh) 5

[4.2 Functional decomposition](#_heading=h.b1ww7ekgigjo) 6

[4.3 Task network](#_heading=h.8l1dmbw4jdyz) 6

[4.4 Timeline chart](#_heading=h.ubh9gmsd3fs) 7

[**5.0 Staff Organization**](#_heading=h.g7e2ktvri374) **8**

[5.1 Team structure](#_heading=h.ll3i99x2hdi7) 8

[5.2 Management reporting and communication](#_heading=h.gvewq2tzw8dg) 8

[**6.0 Tracking and Control Mechanisms**](#_heading=h.l9kra6wqaj9z) **8**

[6.1 Quality assurance and control](#_heading=h.2rtt8n6l8vgr) 8

[6.2 Change management and control](#_heading=h.dusak94y5dhl) 8

[**7.0 Appendix**](#_heading=h.x68hhlqn6u8n) **8**

# 

# 1.0 Introduction

## **1.1 Project scope**

Inputs:

* A financial audit form in a readable PDF file

Processing Functionality:

* Read in file values, starting at the top of the file
* Store found values within predefined user interest; otherwise, ignore

Major Outputs:

* An organized CSV file based on extracted data values

## **1.2 Major software functions**

Core features:

* PDF file selection
* Data parsing
* Data extraction to CSV

## **1.3 Performance/Behavior issues**

* Audit PDF files must be readable, i.e., “searchable” and not just scanned images
* Program must export to a CSV file

## **1.4 Management and technical constraints**

* Various documentation deadlines
* Hard-set prototype delivery date; due by end of Winter 2021 semester
* Hard-set delivery date; due at end of Summer 2021 semester
* Developer training in new languages
* Developer adapting to previous team’s code

# 2.0 Project Estimates

## **2.1 Historical data used for estimates**

University projects:

* Typical CIS 350 Project: 250-300 Lines of code, 20 hours per person to complete (including documentation, designing, coding, and testing)
* Previous team’s product: estimated 4,000 lines of code, 400 hours total
* *Source*: R. Klaus, J. MacFadyen, and M. Patel, “Software Project Plan,” 25-Feb-20AD.

## **2.2 Estimation techniques applied and results**

### **2.2.1 Estimation technique *m***

1. COCOMO
2. Function Points

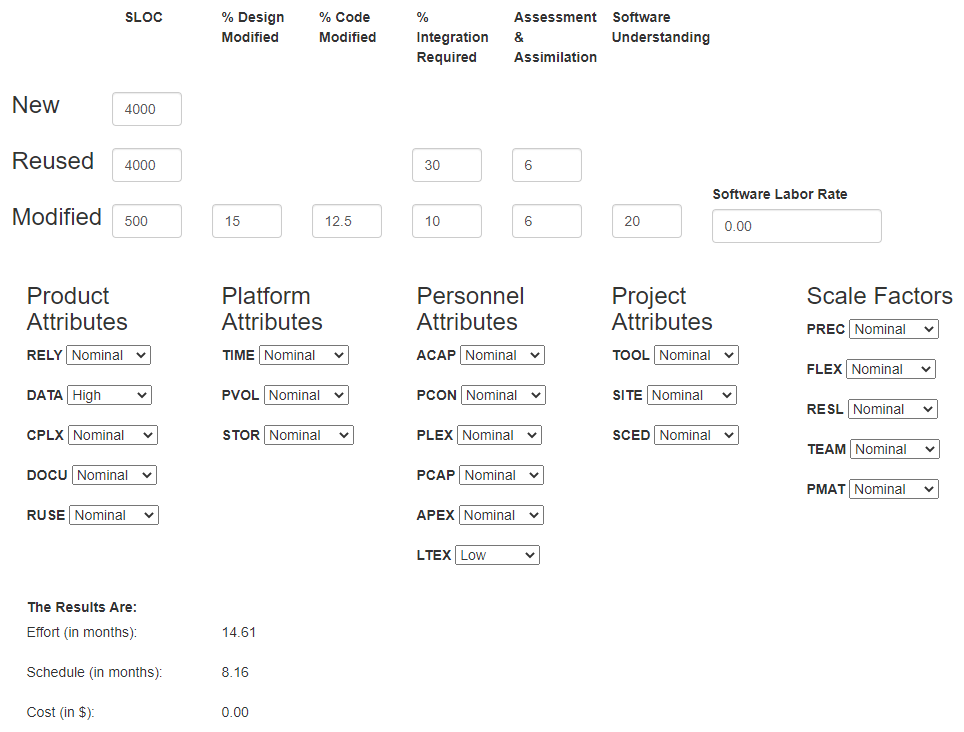
### **2.2.2 Estimate for technique *m***

* COCOMO
  + Semi-detached
  + Estimating 4,000 more LOC
  + Thus,
* For basic model, have:
  + Effort man-months
  + Schedule months
  + Productivity DSI/MM
  + Avg. Staffing FSP
* Function Points
  + Since inputs, outputs, etc. remain the same as the last iteration of the project, function points will be unchanged.
  + For reference, the previous team’s estimation is provided below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Complexity** | | |
| **Low** | **Average** | **High** |
| **EI** | 3 | 4 | 6 |
| **EO** | 4 | 5 | 7 |
| **EQ** | 4 | 5 | 6 |
| **EIF** | 5 | 7 | 10 |
| **ILF** | 7 | 9 | 10 |

|  |  |  |
| --- | --- | --- |
| **Category** | **How Many** | **Complexity** |
| **EI** | 1 | 6 |
| **EO** | 2 | 7 |
| **EQ** | 3 | 6 |
| **ILF** | 1 | 10 |

* 1\*6 + 2\*7 + 3\*6 + 1\*10 = 48 Function Points
* 1 Function Point = 8 hours of work
* Estimate = 48 \* 8 = 384 Hours
* *Source*: R. Klaus, J. MacFadyen, and M. Patel, “Software Project Plan,” 25-Feb-20AD.
* COCOMO 2



## **2.3 Reconciled Estimate**

* The final cost is estimated to be $0.00, provided no additional resources are needed.
* The effort required for the project is estimated to be about 14 months of effort.
* The time/schedule estimate is around 7 months.

## **2.4 Project Resources**

* Developers
* Client
* Python
* GitHub
* Computers
* Michigan Treasury audit archives

# 3.0 Risk Management

## **3.1 Project Risks**

* Development risk
* Client risk
* Time risk
* Quality risk
* Update risk
* Tech risk
* Covid risk
* Online communication risk

## **3.2 Risk Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Risks** | **Probability** | **Impact** |
| Developer Risk | Lack of experience/knowledge | 50% | 1 |
| Update Risk | Update program failure resulting in same or worse product | 25% | 1 |
| Quality Risk | Quality doesn’t meet client’s requirements | 15% | 1 |
| Client Risk | Client stops contacting us | 10% | 2 |
| Time Risk | Limited time to work on project | 15% | 2 |
| Tech Risk | The client doesn’t give us all the necessary requirements | 20% | 2 |
| Development Risk | The client doesn’t give us all the necessary requirements | 10% | 3 |
| Online Communication Risk | We cannot meet in person | 30% | 3 |
| COVID Risk | One of our team members gets COVID | 5% | 4 |

## **3.3 Overview of Risk Mitigation, Monitoring, Management**

**Developer Risk**

Much of this risk can be avoided if the team does the appropriate research before development begins. The monitored condition must be how much team members are facing difficulties while coding. If lots of difficulty is faced, the team members are expected to seek help to stay on track.

**Client Risk**

This risk can be minimized by scheduling regular meetings with the clients to make sure both sides are on the same page. The key condition to be monitored is how often the meetings are scheduled. The earlier all needed information is obtained from the client, the better.

**Time Risk**

Good time management is key for proper development of the application. The project deadlines must be closely monitored to prevent the team from falling behind the schedule.

**Development Risk**

Proper requirements collection by the team prior to development is very important. Also, the team must make sure that all necessary resources are available for the project.

**Quality Risk**

To ensure the project meets the client’s quality standards, a prototype must be developed to obtain proper feedback about the direction of the project development. Error-checking, unit testing, and program updates are necessary for the program’s quality.

**Update Risk**

Proper study of the previous team’s code is important to understand whether the program will be an update to the older one or will be written from scratch.

**Tech Risk**

The team must make sure not to use code written by other programmers, or design ideas from other people. This ensures no plagiarism is done and all intellectual property is accounted for.

**COVID Risk**

Team members are responsible for minimizing the COVID-19 risk by making sure minimal physical meetings are scheduled both internally and with the clients.

**Online Communication Risk**

Team members must take into account the possibility that online communication can sometimes be unreliable or go through technical problems unexpectedly.

# 4.0 Project Schedule

## **4.1 Project task set**

The process model our team is following is agile with a scrum framework. Scrum is an agile framework that goes through the steps of developing, delivering, and maintaining products emphasising on software development. The model we are following is agile because the team will be working through a full software development life cycle.

Task sets:

* Planning:
  + SPMP
  + RMMM
  + SQA
  + Revising client and previous team resources
  + Updating versions
  + Gnatt Chart
  + Product Backlog
  + Setting and following deadlines
* Designing and Modeling:
  + ERD
  + SRS
  + Software Design
  + User Interface Design
  + Use Case Diagrams
  + Functional Decomposition
  + Algorithm Design
* Development:
  + Coding, programming, and updating
  + Prototype Implementation
  + Testing
  + Final Product Implementation
  + Releasing

## **4.2 Functional decomposition**

**Planning**

All documentation required for planning the implementation and requirements are made during this phase. Planning will be done based on the requirements gathered from the clients.

**Designing and Modeling:**

The software’s design process includes drawing out the core use cases, product backlog, UI design, and any planned algorithms. It prepares the team for the implementation of the program.

**Development**

This phase includes the implementation of the program. Coding tasks will be split among team members. After the first draft of code is complete, unit testing will take place, and all overlooked bugs and logical flaws will be ironed out for the best possible user experience.

## **4.3 Task network**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Duration (days)  (estimate) | Predecessor Task | Successor Task | Percentage Complete |
| SPMP | 8 | SRS | Design | 100 |
| RMMM | 8 | SRS | Design | 100 |
| SQA | 8 | SPMP, RMMM | Presentation | 0 |
| ERD | 8 | None | Presentation | 0 |
| SRS | 14 | None | Presentation | 50 |
| Use Case Diagrams | 14 | SRS | Implementing Design | 100 |
| User Interface Design | 14 | Use Case Diagrams | Implementing Design | 0 |
| Prototype Implementation | 20 | SPMP, RMMM, SRS, Diagrams | Presentation | 0 |
| Final Product Implementation | 30 | Prototype Implementation | Testing | 0 |
| Releasing | 7 | Final Product Implementation | Presentation | 0 |

## 

## 

## 

## 

## 

## 

## **4.4 Timeline chart**

|  |  |  |
| --- | --- | --- |
| Phase | Planned Completion Date | Deliverables |
| Gather Requirements | 2/09/2021 | * Client Meetings * Team Meetings * Traceability Matrix * Gnatt Chart * Product Backlog |
| Project Planning | 3/3/2021 | * Use Case Document * SPMP * RMMM * SQA * Check with Clients |
| Requirements Analysis | 3/17/2021 | * SRS * Requirement Prioritization * Check with Clients |
| Design | 3/29/2021 | * User Interface Design * Use Case Diagrams * Algorithm Design * Software Design * UML * ERD |
| Implementation | 7/08/2021 | * Prototype * Coding and Programming * Updating Previous Software * Dependencies * Finish Priorities First * Check with Clients |
| Testing | 7/30/2021 | * Unit Testing * Sample Data Testing * Debugging * Analysis of Data * Bugs reworked |
| Deployment | 8/20/2021 | * Presenting Final Product * Releasing Final Product * Giving Final Product to Client |

# 5.0 Staff Organization

## **5.1 Team structure**

The team is using agile egoless team structure. Each team member has broad skill sets and expertise.

Role Definitions

**Ahmed Mawari**

Documentation, senior coder ,Communicator , Testing

**Adham Abdalla**

Documentation lead , coder, Communicator, Testing

**Hasan Alameh**

Documentation, SQA Lead , Communicator, coder

**Tanis Daniels-Wanamaker**

Documentation, senior coder ,Communicator , Testing

## **5.2 Management reporting and communication**

**Mechanisms for Progress Reporting**

Progress is communicated via email. All files sent to team members are sent via Google Drive shared folder. These communications are done informally, unless special documentation of progress is required.

**Mechanisms for Inter/Intra Team Communication**

The team conducts weekly voice meetings via Discord to get task updates and ask questions that may not be answerable via text.

The client is contacted via email for simple questions and via zoom video conferencing for updates and further questions.

# 6.0 Tracking and Control Mechanisms

**6.1 Quality assurance and control**

Multiple testing stages, thorough design phase and use cases.

usage of GitHub repository for code reviews and unit testing of each feature and final product quality assurance.

**6.2 Change management and control**

All changes will be communicated prior on discord, and we will use a GitHub repository to track when and where changes are made.

Revisions primarily handled by team lead and all approval and final decisions handled by client.

**6.3 Tools**

GitHub and Use Cases: To keep clear traceability and quality of code.

Lucidcharts and Enterprise Architect: Creation of diagram-based artifacts

Google Docs and Google Sheets: Version control of artifacts

Major version control: Canvas

# 7.0 Appendix

References:

* R. Klaus, J. MacFadyen, and M. Patel, “Michigan Local Government Data Parser,” 13-Aug-20AD.
* R. Klaus, J. MacFadyen, and M. Patel, “Software Project Plan,” 25-Feb-20AD.