Current Deliverable Milestone:

Requirements Analysis

Project:

Mountain Lion Detection System

Team:

The Life Comrades

Due Date of Current Milestone:

22 Sept 2022

Executive Summary

Contents Page

Executive Summary **1**

Contents Page **1**

Signature Page **2**

Project Plan **3**

- Project Team 3
- Environment 3
- ICSM Common Case 3
- Methodologies 5

Project Development Estimate **6**

- Software Size 6
  - Detection Program 6
  - Database Management System 6
  - User Interface 7
- Schedule 8
- Total Estimated Hours to Execute Tasks 8
- Life Cycle Model 8
- Third Level Development Tasks 9

Requirements **10**

- Summary 10
- Performance-Related Requirements 10
- User Cases 11
  - Case 1 - Flow of Noise Detected 11
  - Case 2 - Control Program 11
- Implementation & Testing 14
- Requirements Summary 14
  - Initial 14
  - 2nd Meeting 15
  - 3rd Meeting 15
  - Final number of total requirements: 15

Signature Page

Lewis Gibney (PM): Lewis Gibney Date: 22 Sept 2022

Halah Salman: Halah D. Salman Date: 22 Sept 2022

June Vincent Magat: June Magat Date: 22 Sept 2022

Project Plan

Project Team

The team, “The Life Comrades”, currently has 3 team members following a group fusion and people dropping the class. Each member has specific roles;

- Lewis Gibney - Project Manager, UI Designer - Incharge of making sure the team works cohesively, whilst also utilizing past experience to implement an industry grade UI.
- Halah Salman - Test Manager, Programmer - Responsible for assuring proper testing through each of the different components and also aiding in implementing the system design.
- June Vincent Magat - Chief Programmer - Incharge of implementing the system design in a robust, reusable and reliable manner.

Environment

After digesting the project, it was understood that it roughly followed a CRUD application structure. Applications that have a requirement to manage persistent data often use DBs (databases). DBs are a key technology within the web-sphere. Therefore, due to the necessity of a DB within the project and integrity of the technology within the web, we are going to use a modern and up-to-date web stack to implement the project. This means the main codebase of the project will be generated using HTML, CSS and JavaScript.

Furthermore, we will be using React as it provides an open source well maintained codebase that allows us to get the most from a web application. On top of react, we will use Bootstrap to easily manage the UI elements whilst also achieving a modern aesthetic. We will require a DB to store the data, at this preliminary stage we plan to use MongoDB self hosted as it’s a very popular tried-and-tested platform.

The project will also require us to detect specific animal noises from detection sensors, for this we will use the Raspberry Pi platform coupled with python for our programming needs. Raspberry Pis have been chosen due to their relative cheapness and impressive functionality.

ICSM Common Case

The ICSM is a risk driven framework or meta-model that is built on principles and supports the creation of life-cycle processes based on the features, limitations, and risks associated with a specific project or program. The ICSM common cases have been developed to show the users how to use the framework. Because the goal of the ICSM is to identify the risks in software development these common cases are essentially situations that present risks at the various stages of the development. The common cases we will be considering for our project based on our project are:

Project	Cases	About the Component	It’s Use	Rationale
Mountain Lion Detection Application	Software application or system	Database management system, command and control/sensor processing system.	For our project we will have multiple databases to store the information, send and retrieve it.	-safety/security of critical components -for selecting and implementing solutions
Raspberry Pi	Software-intensive device	A piece of equipment that is developed for a special purpose and has significant features provided by the software.	For our Project we will be using Raspberry Pi and it’s Raspbian OS System to implement it.	-to implement the detection system and transfer data -meet certain price point -upgrade and fix problems
Multi Sensor, Laptop/s/phones (We aren’t developing the hardware this just shows all the components needed for the system)	Hardware platform	Hardware platform	A crucial part of the project requires the display to be able to use sensor/s to implement it/run it.	-to implement the system -to display the output of the system
Detection Management	System of System(SoS) or enterprise-wide system	The collection of systems we will be using to implement our project, which we can also update and fix.	For our project, we need to get data from the sensor and create a system that works together to recognize, work on, transfer and store the data in it.	-to integrate a set of existing systems -guide and evolve the integration of a set of existing systems
Maintenance	Brownfield modernization	Incremental replacement of old, fragile business systems with COTS products or technology refreshment/up-grading of existing systems.	For our project we will need to be able to fix and upgrade the system from time to time.	-old systems can contain bugs/vulnerabilities. -old systems need new features to keep it popular.

Methodologies

We will be using multiple of these methods which count as part of the ICSM(Incremental Commitment Spiral Model) Software Strategies to respond to the challenges that may arise from designing/implementing the Mountain Lion detection system.

Name	Priority level	Explanation
Architected Agile (Using Jira scrum agile methodology)	Low to Medium	We are using Architected Agile to create a strong software foundation and architecture, then transition to purely Agile Process.
Agile (Using Jira)	Low to Medium	We are using the Agile methods for the software development process to develop the software capabilities

Project Development Estimate

Software Size

Based on our lack of previous experience with writing and implementing large software systems, we expect our actual software size to vary from our estimates. However, with that said, we will try to use prior experience where applicable to estimate the size of the different components, those being; the detection program, the database management system & the user interface. We will focus on function point sizing.

Detection Program

The detection program, currently slated to be coded in python, will be of a rather average level of complexity with ~100 lines of code. The program will be incharge of reading sensor data from the raspberry PIs inputs, performing a calculation to check for a detection & finally returning a JSON holding detection details to the DB when the threshold is met.

Components	Low	Average	High	Details	Score
Internal Logic Files		1		One average python file	10
External Interface Files	1			Audio level details to trigger detection	5
External Inputs			1	Continuous audio signal	6
External Outputs			1	JSON with confirmation of detection	7
External Queries					0
				UFP Count	28

Database Management System

The DB will require code to manage the data as requested in the briefing. The system must be able to edit the DB based on time elapsed and update the DB with new incoming detections, whilst also being able to serve data when called for.

Components	Low	Average	High	Details	Score
Internal Logic Files			1	Code for delivering and managing saved data	15
External Interface Files					0
External Inputs			2	Database and detectors	12

External Outputs			1	Data for re-entry in to database	7
External Queries			1	Data for the user interface	6
				UFP Count	40

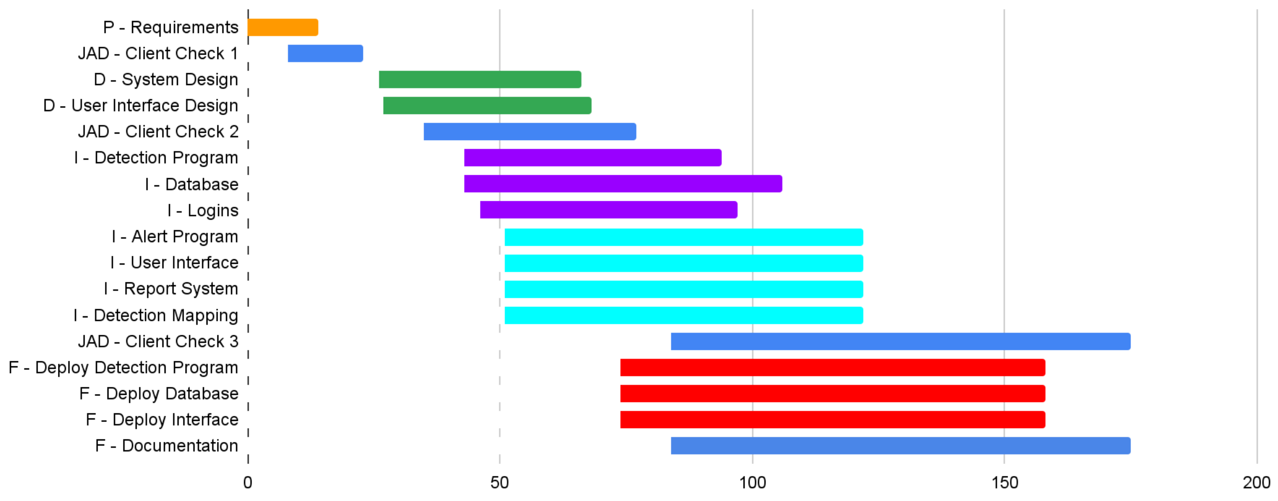
User Interface

The UI will be generated with react, so both our functional code and display code will be housed in the same files. At this early point of development we foresee 3 different pages; a landing/login page, a report page and a graphical report page (where the data is shown across a map). We estimate that the code for this section will be the largest.

Components	Low	Average	High	Details	Score
Internal Logic Files	0	1	2	3 different react page files	40
External Interface Files					0
External Inputs			1	Input from database management system	6
External Outputs					0
External Queries					0
				UFP Count	46

Schedule

We currently have the tasks scheduled to make full utilization of the semester. We start first in the preliminary phase where we take the time to break down the requirements and then start documentation. After completing the documents we go on to have our first client check. Following a positive or otherwise resolved client meeting, we head to the design phase. After designing the system we will have another meeting with the client to ensure they are happy with the initial design, following any alterations we start the implementation phase. The implementation phase has been split up into 2 sub-sections that can be roughly encapsulated as backend (detector, database and login system) and frontend (alert program, UI, report system and detection mapping). After completing the implementation phase we will have a prototype system working in a development environment, we will then show this to the client to ensure they are happy before the final stage. The final stage is to deploy the prototype to a physical hardware environment that is similar to its final form. In this final phase we will also confirm that all the proper documentation has been created.

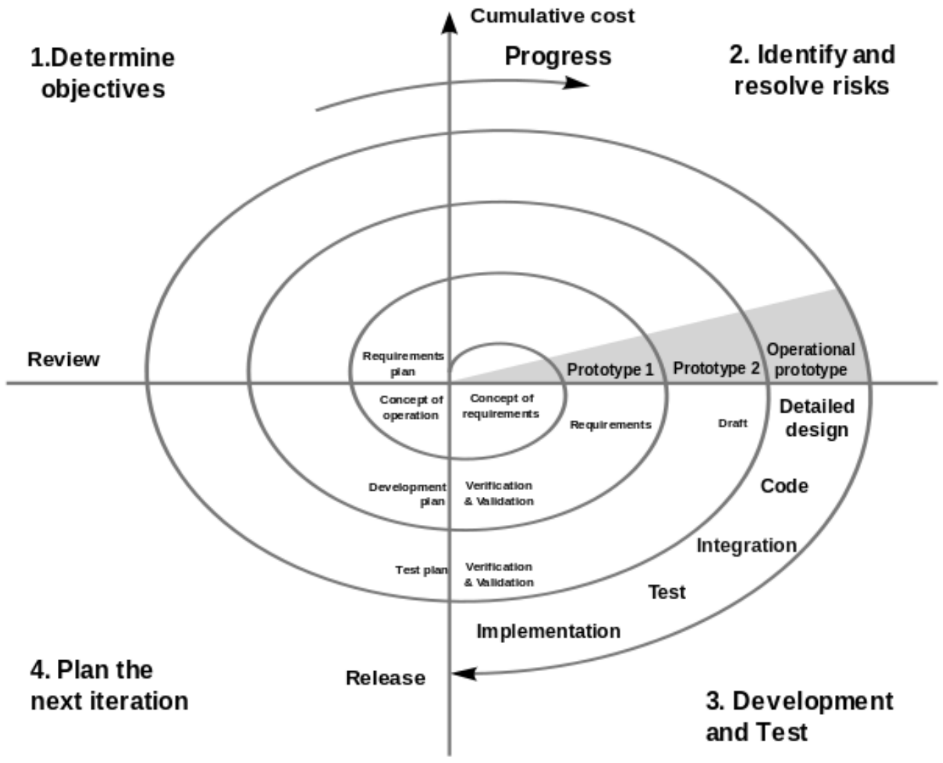


Total Estimated Hours to Execute Tasks

If we extrapolate the time from the schedule, by taking each day on each task as being an hour we can estimate the time the project will take to execute. This total is 213 hours, which should roughly end up being 71 hours per group member.

Life Cycle Model

The life cycle model we choose to go with is the Spiral model due to its unique ability to include a more detailed planning of the project. It is also more compatible with risk analysis.



Third Level Development Tasks

The practical execution of tests happens in the third stage. The simultaneous application of developmental variants leads to this. It involves tasks like creating test cases, writing test summaries, coding, drafting bug reports, and actually running tests as shown above. Some dates may overlap due to the constant testing and implementation of the programm. By the 22nd of September we would have finished the first and we need to work on building a design model and evaluate and resolve the risks for the second phase. Then shift to the third phase. Meaning for the third phase we will need to start working on our detailed design using Jira(Scrum) and Miro. Based on our experience working with different software system projects the third phase more specifically the Development part is the most critical. It is also the most time consuming, mostly due to it being hands-on and actually building the software system. Although integration and testing can be exhausting they can be done within a small time frame.

Task	Date(from-to)
Detailed Design	9/22-10/20
Programming	10/20-11/15
Integration	11/15-11/28
Test	11/28-12/1
Implementation	12/1-12/8

Requirements

Summary

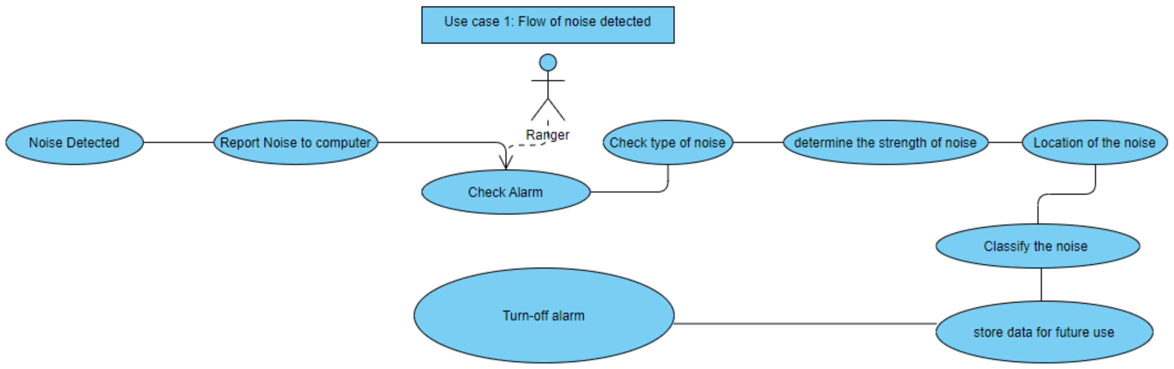
The system is an animal detection system whose main purpose is to send an alert to park rangers if a mountain lion is detected within 5 miles of the detection sensor. The system is designed to distinguish between various types of animal noise and send the noise type to a control computer located in the park ranger station. The control program will display several types of reports upon request. Reports showing the date and classification of the noise whether it be definite, suspected, or false. It will also show the specific location of the detection from the sensor location. This system will have a graphical report showing detections within 2 miles of the park, and will also have an option where it will show the classifications made by the rangers.

Performance-Related Requirements

There will be a detection sensor throughout the park, the sensor will transmit the recorded noise to the control computer to alert the rangers. Park rangers will determine the type of noise, and the system will display the loudness of the noise and its location. This information will be stored for future access. Only the park ranger who controls the computer can turn off the alarm.

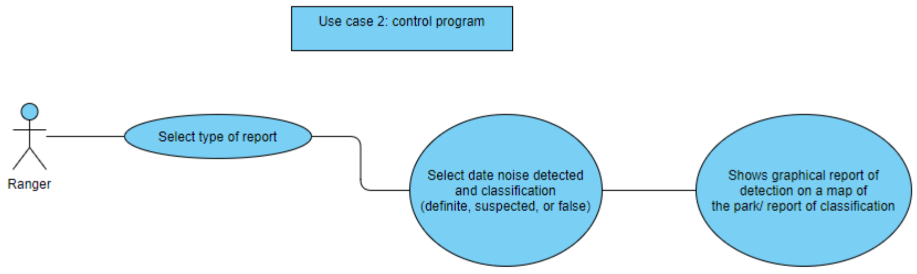
User Cases

Case 1 - Flow of Noise Detected



In the first case, the noise detected by the sensor will be sent to the control computer to trigger an alarm, alerting the ranger (user) to check the recorded noise. The user then classifies the noise as definite, suspected, or false based on the noise detected. This information will be stored in the database as a report for the second use case. Only authorized users (Rangers) are allowed to have access and the ability to turn-off alarms.

Case 2 - Control Program



Use case 2: control program
Ranger
Select type of report
Select date noise detected and classification (definite, suspected, or false)
Shows graphical report of detection on a map of the park/ report of classification

The purpose of the second use case is to display the previous reports of authorized personnel (Park Rangers). Rangers can select the time of the date when there was a detection of noise and the classification accompanying it. This report will show the sensor that detects the noise on the map and there will be a graphical display of this report on the screen.

Discussion of requirements analysis:

Functional Requirements	Allocation of Requirements	Increment/evolution	Requirements to be Implemented for Increment/evolution
Place noise detection sensors within an area of 5 square miles	Hardware	Evolutionary	Will require the use of a sensor that transmits sound to the software intensive system.
Device should be able to detect various types of animal noises	System Software	High Priority- must be implemented first	Will require a working device that detects noise integrated with a program that determines the loudness of the noise to trigger the alert.
Alert messages be sent to the controlling computer based on the type of animal noise detected and the strength of the detected animal noise	System Software	High Priority- must be implemented first	Part of the Software intensive system functionality.
Alert messages will contain the type of noise detected, the strength of the detected noise, and the location of the detected noise to within 3 meters.	System Software	High Priority- must be implemented first	Part of the Software intensive system functionality.
The control program on the controlling computer will sound an alarm whenever An alert message is received from the animal detection system. The alarm will Continue until the ranger turns it off. Once the alarm is turned off, it will not sound again until another, separate noise is detected at a different location.	For us the alarm will be part of the application so it will be a Software Application.	Mid Priority - will be implemented next after highest priority has been proven functional	This is mainly through Web-application which will be designed to set Alarm whenever alert is received.

The controlling computer will save all mountain lion alerts received within the last 30 days. It will save a summary of alert information for data older than 30 days, but received within one year.	Software Application	Mid Priority - will be implemented next after highest priority has been proven functional	This will be part of the Web-application functionality because the Web application is signaling to save the data into the database.
The control program on the controlling computer will allow the ranger to classify each alert as definite, suspected, or false, indicating the probability that a real mountain lion was detected.	Software Application	Mid Priority - will be implemented next after highest priority has been proven functional	This will be part of the Web-application functionality to perform specific tasks such as classification.
The control program will allow the ranger to request several Reports: A report showing al l mountain lion detections by date detected and by classification (definite, suspected, or false) A report showing all mountain lion detections at a specific sensor location. A graphical report showing detections on a map of the park and areas within 2 miles of the park. A report showing detection classifications by ranger.	Software Application for performing the tasks System Software for Databases	Low Priority- last to be implemented	Part of the web-application to display landing page, login page and report page. Will be implemented last once all the requirements has been proven functional.
The system should be developed in a way that would allow it to be easily reconfigured for other parks in the State of California.	Software Application, System Software and Hardware.	Evolutionary	This will include all systems because we have to keep up to date with the latest technology, fix security issues that may arise and add more features.

Implementation & Testing

First, we will test the noise detection sensors, this will be achieved by integrating our programs into raspberry pi hardware. Once, that's been established as operating correctly. We will then move on to the Alert messages testing to see if an alert is being sent after the sensors have detected noise. We would then move on to making sure that controlling computers are receiving these alerts, it is important because this is how the rangers will be able to classify the noise detected and log the information to be sent to the database. Lastly, we will need to test if the information are being stored in the database and the graphical display is showing the right information when requested by the Ranger.

Work Done on Requirements Summary

Meeting #	Date & Time	Requirements	Notes
Initial	05/09/2022 @ 11:00 A.M. - 12 P.M.	Hardware	Raspberry Pi, Other Hardwares, etc.
2nd Meeting	08/09/2022 @ 6:15-6:35 P.M.	System Software	System requirements i.e. database
3rd Meeting	16/09/2022 @ 11:00 A.M. - 12 P.M	Software Application	Discussed software application and requirements.

Initial

In our first meeting as a group, we discussed the hardware requirements for the project. We decided to use a Raspberry Pi to implement data transfer between the detection system and the database. We also agreed that our group will use this as a sensor device that detects noise to assist in the implementation of the detection system. Lastly, the computer that will display the output of the system.
Number of initial Requirements discussed: 3

2nd Meeting

Now that the hardware requirements have been established. In our second meeting, we discussed the overall design of the product. Our group decided that we will be needing a database for our log system to store the previous detection's information. To do this, we all agree that we need to allocate most of our efforts to the detection system; our alert program.
Number of requirements discussed: 2

3rd Meeting

Finally, because this program needs a screen. We discussed the need for a UI to display the landing page, login page, and report page. Initially, we wanted to have a registration system within the log-in page, as only authorized personnel would have access to alerts and reports. Ultimately, we all agreed that there was no need for a registration system, as we only needed to implement an admin page for supervisors to add other workers to the system.
Number of requirements discussed: 4

Final number of total requirements

Overall, there are 9 total requirements discussed in our analysis, which are required for our decision to move on to the top-level design for now. A detailed summation of this requirements can be seen under "Discussion of requirement analysis" section of this write-up.

Project Notebook Contents – One per Team

1. Cover Page identifying Deliverable Milestone, Project, Team, and date of report
2. Signature page contains the signature of all team members contributing to the project notebook
3. Project plan contents
    a. Project Team – Roles and responsibilities for each member—must include PM & Test Mgr
    b. Development Environment (and Target Environment if different)
        i. Platform/operating system
        ii. Implementation language(s)
        iii. Database application
        iv. Rationale for selection
    c. ICSM Common Case to be used for development and rationale
    d. Methodologies/techniques to be used in development and rationale
4. Project Development Estimate
    a. Software Size *This will be updated in subsequent deliverables*
        i. Provide software size estimates using the one of the three methods discussed in class: lines of code, function points, or object points
        ii. Estimates should be done by software "component" and identify the associated complexity of each.
    b. Schedule (OpenProj, MS Project, Visio, MS Excel, Google Sheets, …)
        i. List of third level development tasks (based on life cycle model selected)
        ii. For each 3rd level task, identify planned start date, duration of task, and any predecessor tasks
        iii. Identification of critical path
        iv. Progress to date with respect to the planned schedule
        - *Actual measures: Size, labor hours, schedule in final deliverable*

Note - Schedule to include the following for each major activity: lead person, subtasks, relationships between subtasks (e.g., predecessors/successors), and critical path

5. Estimate of total number of labor hours to perform all tasks in the schedule *This will be updated in subsequent deliverables*

6. Requirements: *This will be updated in subsequent deliverables – initial delivery should have (a) and only a high level (b)*
    a. Statement of Scope of Effort/Understanding of Problem

        i. A paragraph or two summarizing the overall functionality to be provided by software system
        ii. A few paragraphs describing performance-related requirements for the software system
        iii. Use Cases (UML diagram format)

   b. Discussion of requirements analysis (*Note: complete details of requirements, requirements allocation, and mapping to increments expected to be in Project Notebook and more detail developed for each subsequent delivery*)
        i. Table that summarizes
            1. Your requirements (by functional areas, features, or use cases)
            2. Allocation of requirements to hardware, system software, application software, user process
            3. Requirements to be implemented in each increment/evolution
        ii. Discussion of completeness, consistency, testability, etc. of requirements and any changes that were required to address these types of issues
        iii. Quantitative summary of requirements at end of requirements analysis
            1. Total number from initial requirements document by functional area
            2. Number of requirements added, deleted, or changed during requirements analysis activities
            3. Total number of requirements going forward into top-level design

7. Design:
   a. Top level architecture diagram for subsystem, including rationale for architecture style selected—diagram must be specific to your project. This means that your components must have names meaningful to your application. Discussion should describe the purpose and scope of each component as well as the connectors between the components.
   b. Database Design – Entity-Relationship Diagram (ERD) showing tables, attributes and relationships between tables. Also includes a database dictionary that defines all of the tables, fields/attributes, primary keys, foreign keys, indexes, and relationships.
   c. Requirements mapping to architecture components (this should be provided in your separate requirements matrix/repository).
   d. User interface prototype – include images/drawings used in designing the UI
   e. Detailed design
   f. Informal notes related to design activities
   g. Design review notes (top level, detailed design - optional*)
8. Code and Test:
   a. Code review notes (optional*)
   b. Unit test plan
   c. Unit test review notes (optional*)
   d. Unit test/retest results
   e. Integration test plan

   f. Integration test review notes (optional*)
   g. Integration test/retest results

9. Risk status/ Areas needing further analysis/questions still not answered and plans for getting risks/issues resolved. *This will be updated in subsequent deliverables*

10. Independent System Test: *each team will build a – c for your system, include d from the team that tested your system*
    a. Test Plan/Strategy
        i. Plan for testing project requirements to include
            · Test method for each requirement
            · Test tools and data required for each test
        ii. Informal notes related to test planning activities
        iii. Test plan review notes (optional*)
    b. Test Procedures
        i. For each test procedure/case:
            1. Description of test procedure/case steps
            2. Requirements tested by each procedure/case
            3. Inputs for each test procedure/case
            4. Expected results
            5. Pass/fail criteria
        ii. Test procedure review/inspection notes (optional*)
    c. Test Tool Development (if applicable)
        i. Test tool design
        ii. Test tool design review/inspection notes (optional*)
        iii. Test tool testing results
    d. Test Results Report
        i. List of tests executed and results
        ii. Defect discovery profile
        iii. Defect closure profile

11. Timesheet for all team members *This will be updated in every deliverable*

* At least 5 optional items (7, 8, and 10) are expected for an overall grade of A on teamwork