SOFTWARE REQUIREMENTS AN ENGINEERING

Energy management system

Prepared by

Hemanth Boinipally

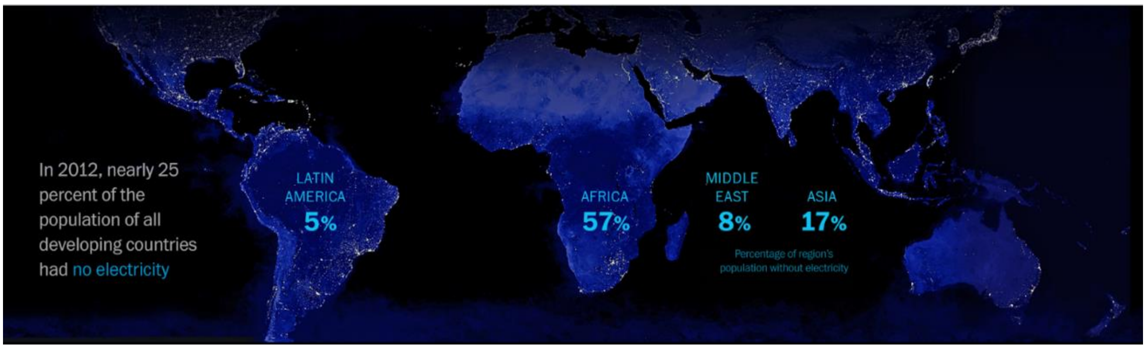
Ashish Reddy Yeruva

Diya

Contents

1 Mission statement

- There is a problem with supplying electricity to all parts of the world; to address this, we are developing a project that will allow us to use renewable energy to supply it to various parts of the world at a low cost and with ease of access; we are considering homeowners who have unused energy as vendors and homeowners who cannot access electricity as lenders; the vendors can lend the power from solar panels, generators, and other sources.
- Using solar panels would assist to reduce air pollution, water usage, reliance on nonrenewable energy sources, and improve long-term human health.



2 stakeholders

Internal Stakeholder:

- An internal stakeholder is a person or group who is affected by a business process.
- An internal stakeholder is a person or group who genuinely cares about a project.
- What is the function of a stakeholder in the discussion about An internal stakeholder is said to be employed in the decision-making process.
- This procedure involves huge corporations, government entities, and non-profit organizations.

External Stakeholder:

- The primary job of internal stakeholders is to invest in or exit a business. External stakeholders, on the other hand, have little influence over the company's operations.
- Do not participate in any company's internal matters, as you may have guessed from the names of external stakeholders.

- **Internal Stakeholders**: -

- UI/UX Developers: I/UX designers oversee designing and implementing all of the experiences that a user has when dealing with a digital tool, such as a website. The user interface/user experience designer will collaborate closely with our marketing team and designers to guarantee seamless web/mobile design and the successful implementation of UI/UX best practices and principles across all our digital platforms. UI/UX designers oversee designing and implementing all of the experiences that a user has when dealing with a digital tool, such as a website. The user interface/user experience designer will collaborate closely with our marketing team and designers to guarantee seamless web/mobile design and the successful implementation of UI/UX best practices and principles across all our digital platforms.
- Q/A Engineers: Quality assurance is the primary function of QA. A QA engineer is responsible for enhancing software development processes and preventing production problems. In other words, they ensure that the software development team is doing things correctly. The job description of a QA engineer includes a variety of responsibilities. Checking if the product complies with the requirements, assessing risks, Planning ideas to improve product quality, Planning tests, Analyzing the test results
- Developers: developers have to do works like Software research, design, implementation, and management, New program testing and evaluation, Identifying areas for improvement in existing programs and then implementing these improvements, Writing and deploying effective code, assessing operational feasibility, Creating methods for quality assurance, Putting software tools, processes, and measurements in place, Existing systems must be maintained and upgraded., User education, Collaboration with other developers, UX designers, business analysts, and systems analysts
- Project Managers: Project managers are in charge of planning and supervising projects to ensure they are finished on time and within budget. Project managers plan and allocate project resources, create budgets, track progress, and keep stakeholders updated throughout the process. All of this is done within the framework of a company's goals and vision. Project managers are needed for a wide range of initiatives, including construction, information technology, human resources, and marketing.
- Business Analysts: A business analyst is an essential member of any project team. They gather information, document processes, and confirm final documents with users as the primary interaction between users and the project manager.

- **External Stakeholders: -**

- Customers: The customer's role is not only to accept results but also to guide the team, address their concerns, and make sure that they're doing things the way they should.

- End Users: An end user is a hands-on user of a product who uses the delivery on a regular or daily basis. They play an extremely important role in product development. End users provide feedback to developers, which helps to ensure that software products are used by the people who need them.

2 stakeholders

- Government: Government allows citizens to have a voice in how their society is run. It also allows the government to get feedback on the project and make necessary changes. Without Government, Software projects would be carried out without input from those who will be most affected by them.
- Suppliers: Suppliers Are important as the software projects are heavily dependent on the supplies.
- Sales Supervisor: Sales Supervision is important because the software projects should be financially fruitful for the company to survive.
- Marketing Team: Marketing team is an extension of the sales team as the marketing team markets the product for the public display.
- Customer Service Team: Gaining a new customer is as important as looking after an old one, The customer service team takes care of the customers issues and troubleshooting problems.
- Installation Companies: Installation companies diversify the company and help with the reach of the product and organization.
- Component Manufacturers: Component manufacturers are important as they are the source of the hardware which is the muscle of the project.
- Roofers: Roofing software is a type of cloud-based business management software that has been designed exclusively for the roofing industry. It was created to help roofing companies and their various teams effectively and efficiently manage tasks and roofing projects.
- HVAC installers: HVAC installers improve with resource allocation and customer service

Key drivers

Key drivers are the most important aspects influencing a company's or business's performance. A major driver is something that has a significant impact on how well the business performs. It can also provide early warning indicators of poor performance or outcomes. Let's take a look at some crucial driver selection recommendations and examples.

- Number of locations: as the business is about solar energy the manufacturing plants are meant to be opened at every major country like USA, RUSSIA, CHINA, UK, INDIA as we can find skilled and unskilled workers with ease and has population with good literacy rate where they can understand the concept and has the idea about the solar panels. However, our target is to get electricity to the rural areas more so, we will have stores opened near them and we have sales site where people can get our products from any spot in world
- Traffic volume to your business website: Website traffic refers to the number of people who visit a website. Web traffic is measured in visits, sometimes known as "sessions," and is a standard approach to assess an online business's ability to attract customers. When ecommerce initially took off in the 1990s, web traffic was seen as the most crucial statistic for measuring a website's popularity because alternative metrics to gauge online performance did not yet exist. Analyzing a website's performance becomes far more detailed as digital marketers become savvier.
- Effectiveness of the sales team;": Sales effectiveness = output per salesperson. "George bronten following this statement, our sales team has best effectiveness compared to regular industry standards. they did plan to educate rural people of pros of our business and are contacting social media influencers as advertising means due to budget constraints and they would reach corner of the globe very easily
- Number and price of offerings: as we are planning to sell our products all over the world. The pricing would be different for every place. as there are different manufacturing costs in each place. For example, India has low manufacturing cost and labor costs compared to the USA or Canada. So, the selling cost would be low too .as the cost of living is also low in India, we should sell our product accordingly.
- Customer satisfaction: customer plays an important role in the business because he is the destination for any company. Mouth publicity plays a major role in the business .so, we have given out our product to some influential people all over the world and some to some volunteers as beta testers and we have got a great response from them
- Staff turnover: as the company is on a global scale we need a lot of unskilled labor and quite a number of skilled labor. where the unskilled workers' pay range would be between 5,000 dollars to 50,000 dollars and skilled workers' pay range would be ranging from 25,000 dollars to 125,000 dollars. there is sales team who will also be in range of skilled workers

Problem's Fit to Life-Cycle Process:

The reliability and stability of electricity in some parts of the world is under high variance. Our Management System product will help alleviate some of the problems and bring peace of mind to both homeowners and businesspeople.

Tentative Solution:

Our product will be a small device that will need to be installed and connected to the local power grid that will analyze the fluctuation of electricity in order to procure a steady output of power for the building.

Development Strategy:

Key drivers

The software will be developed in an iterative, Agile-like manner. Due to this being a small to medium sized project, and that there may be many currently unforeseen problems or changes, developing the product iteratively will help us overcome the challenges with less wasted effort and time.

Targeted Users:

The intended customer for our solution will be homeowners and small businesses. The reason for this is that for many of these potential customers, having high variance and unpredictability in their power system can be more detrimental than for people with larger capital and more resources. Also, these buildings are more likely to be surrounded by buildings that also have power issues, and diverting excess power and stabilizing the overall grid will be more beneficial to the neighborhood or small community.

Deployment Strategy:

We intend on delivering our product to low importance buildings and delivering to many buildings in the area in order to have a safety net in case some of the devices fail. In this case some of the devices in the neighborhood can take the load temporarily meanwhile a technician/homeowner sees what they can do to fix the problem.

- Our product will allow for the attachment of additional power sources like solar panels. Many homeowners consider putting solar panels on their houses in order to produce more electricity, lower their electricity bills, and to lower the effect main power grid fluctuations have on them.
- Our product must accommodate these modifications in order to be marketable to these people.

Development Environment & Tentative Output:

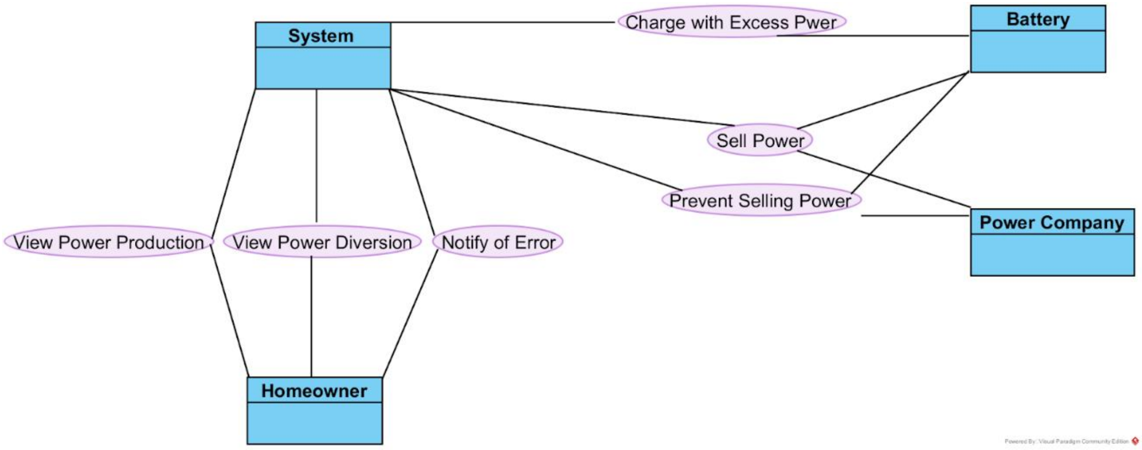
- Our product will be inside a small Linux device that will connect to the building's power grid.
- Our product will be a single device managing one building’s power grid while also being able to communicate to other similar devices.
- The device will be connected to the internet and be able to download and install updates on a set period.
- The user will be able to access a limited management portal for the device through the web. Technicians will have full access to the device and will monitor its health on a periodic basis.
- The device will self report a set amount of information that it will store locally. Periodically it will send the reports to the web service. Users will be able to download their devices reports over the internet from anywhere.

6 use case diagrams

UML Use Cases

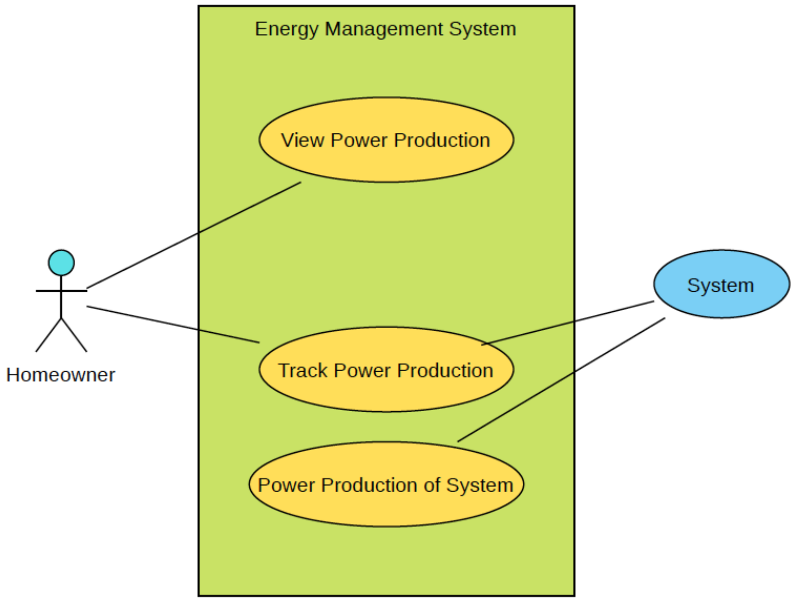
1.
    1. **ID:** EMS01
    2. **Name:** Homeowner can view how much power their system produces
    3. **Actors:** Homeowner, system
    4. **Description:**
        1. System keeps track of how much power is being produced
        2. Homeowner views the information
    5. **Response:** Power production of the system
2.
    1. **ID:** EMS02
    2. **Name:** Homeowner can view how much power is being diverted
    3. **Actors:** Homeowner, system
    4. **Description:**
        1. System keeps track of how much power is being diverted
        2. Homeowner views the information
    5. **Response:** Power diversion of the system
3.
    1. **ID:** EMS03
    2. **Name:** The system can notify the homeowner when an error occurs
    3. **Actors:** System, homeowner
    4. **Description:**
        1. An error occurs in the system
        2. System alerts the homeowner
    5. **Response:** The homeowner is notified about risen errors
4.
    1. **ID:** EMS04
    2. **Name:** The system shall be able to charge a battery with excess power
    3. **Actors:** System, battery
    4. **Description:**
        1. The system detects excess power being produced
        2. System reroutes power to batteries
    5. **Response:** The battery being charged with excess power
5.
    1. **ID:** EMS05
    2. **Name:** The system shall be able to sell power to the power company when battery if full
    3. **Actors:** System, battery, power company
    4. **Description:**
        1. System monitors battery level
        2. When battery becomes fully charged, start drained and selling the power to the power company
    5. **Response:** Power from the battery being sold to the power company
6.
    1. **ID:** EMS06
    2. **Name:** The system shall not sell power once battery as below a set threshold
    3. **Actors:** System, battery
    4. **Description:**
        1. User sets up a threshold below which the battery shall not be drained in order to sell excess power
        2. System monitors battery level
        3. System stops battery drain once threshold is reached
        4. System does not resume drain once battery is fully recharged
    5. **Response:** Battery storage does not fall below a set threshold

6 use case diagrams

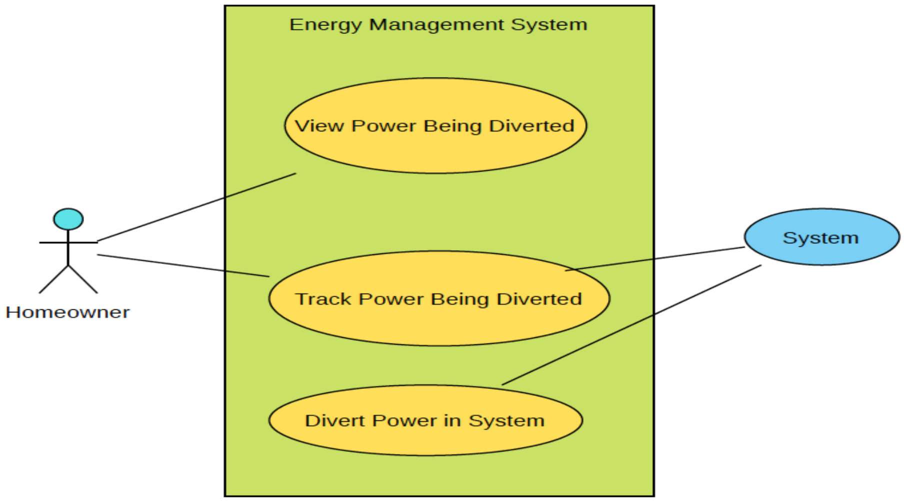


6 use case diagrams

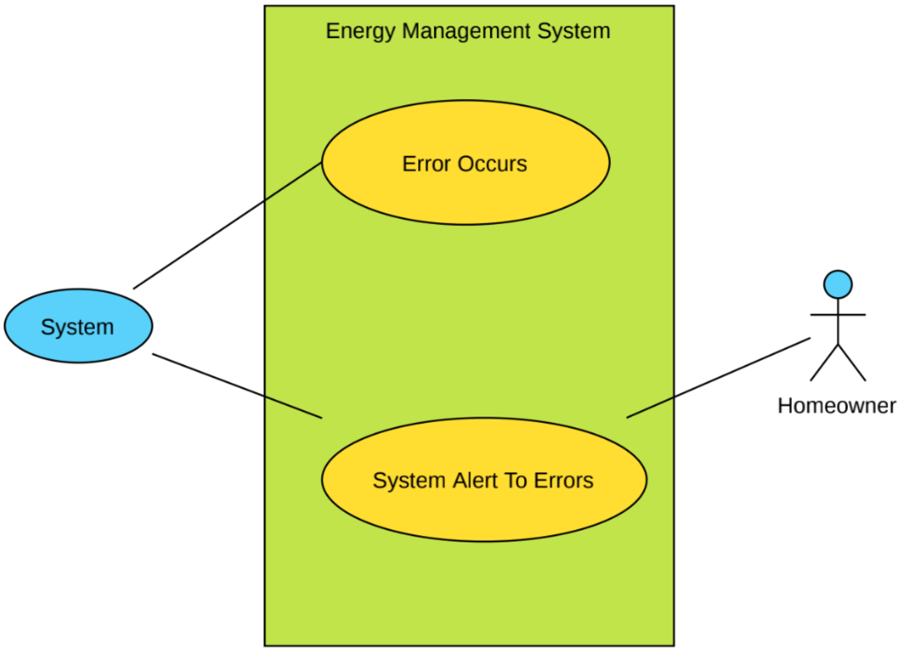
Use Case Modeling



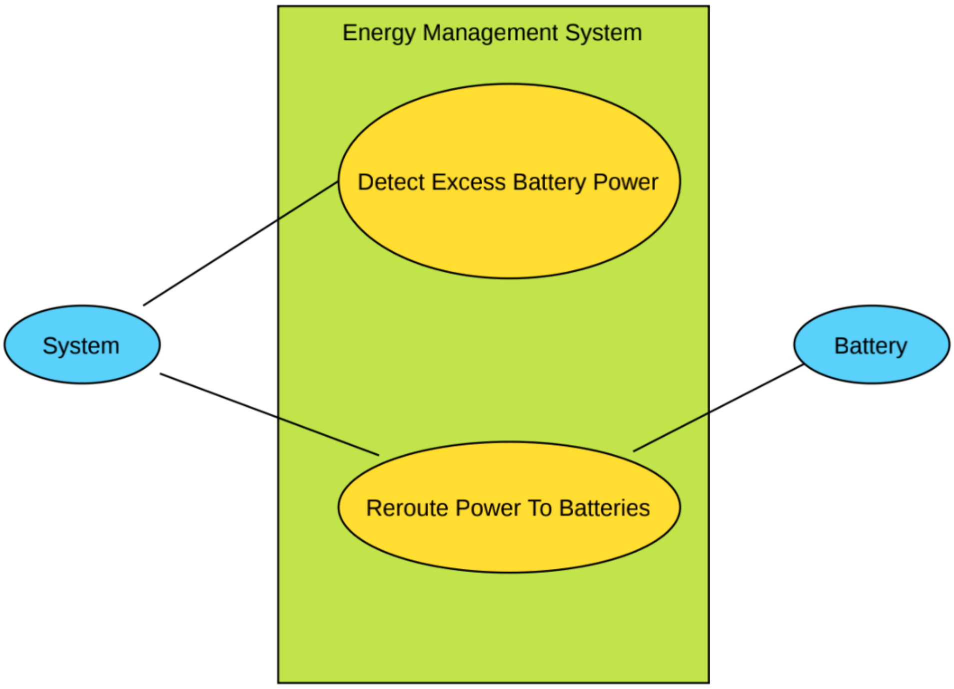
6 use case diagrams



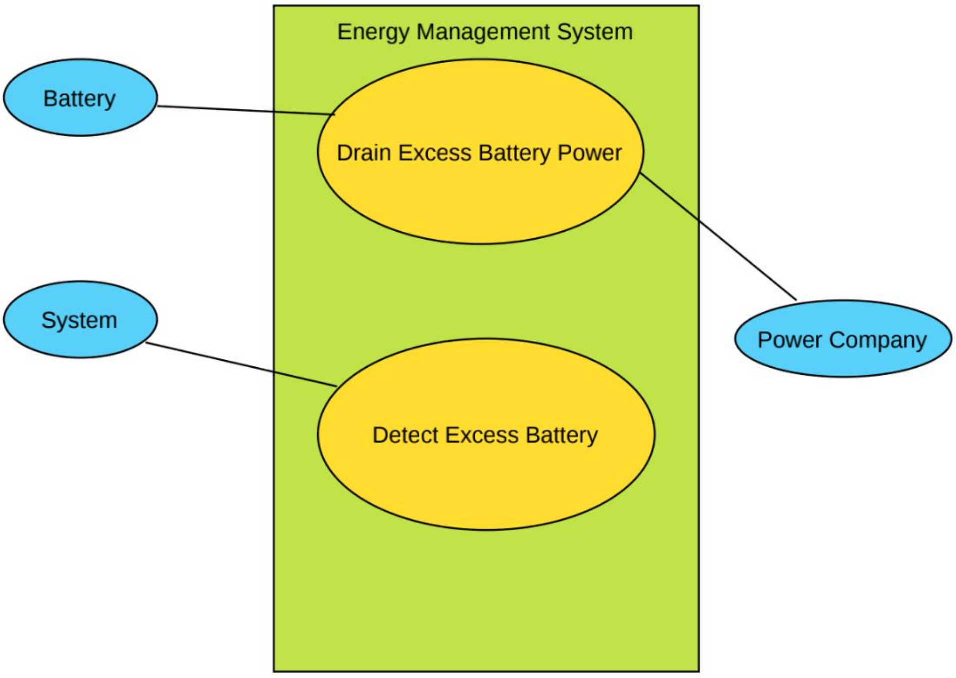
6 use case diagrams



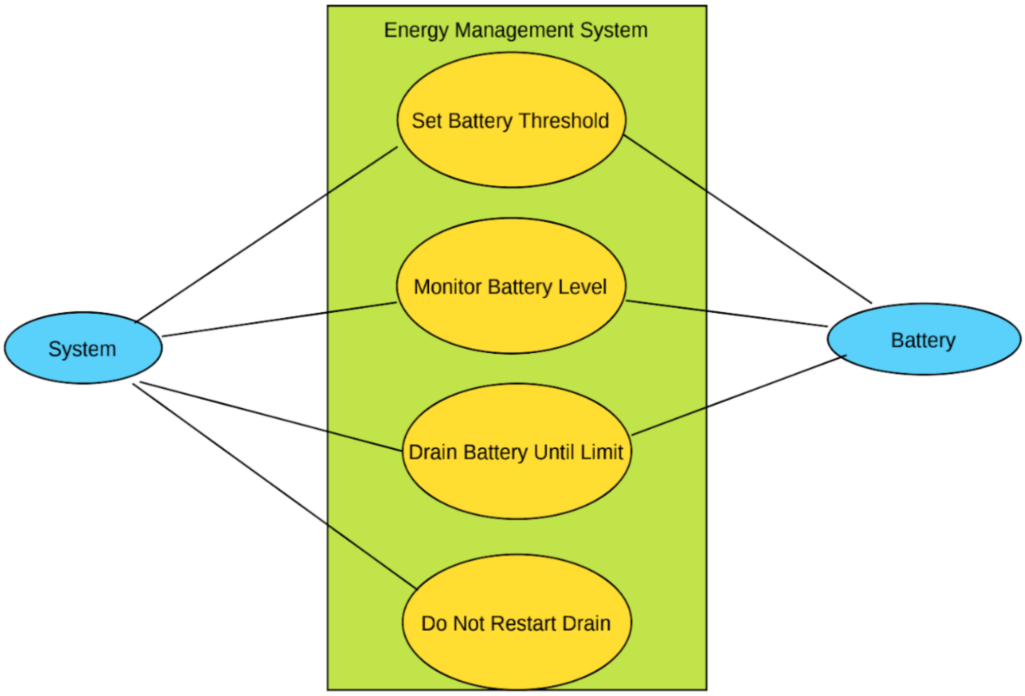
6 use case diagrams



6 use case diagrams

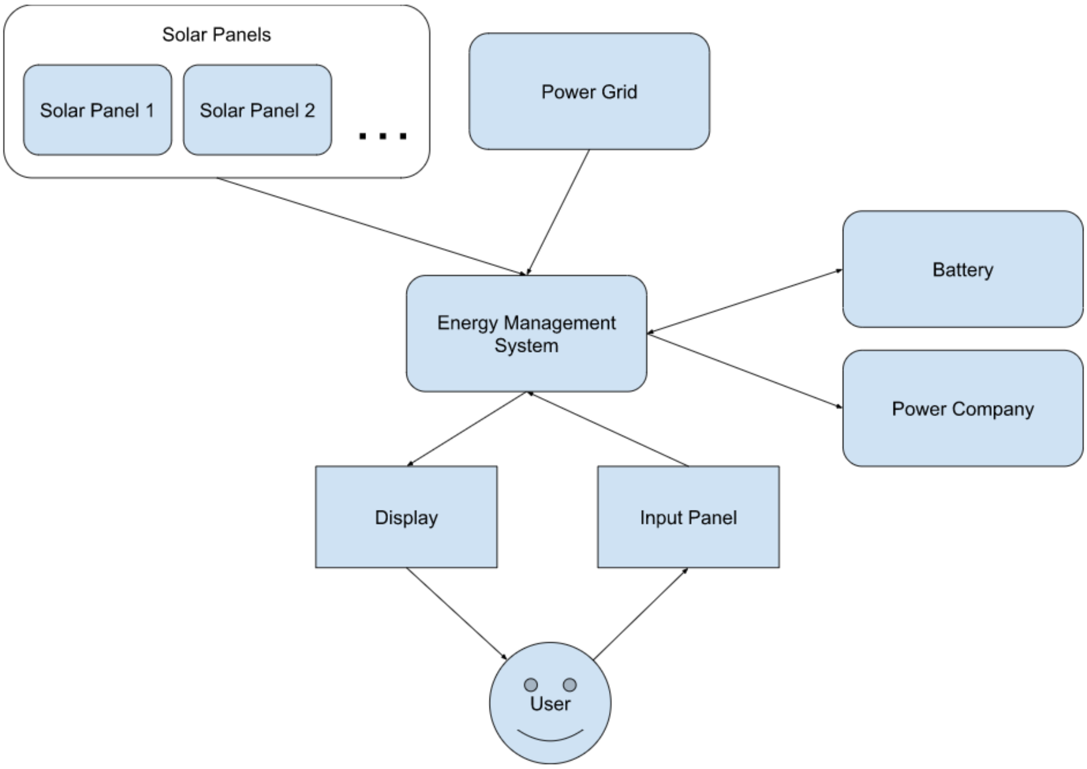


6 use case diagrams



Architecture

6 use case diagrams



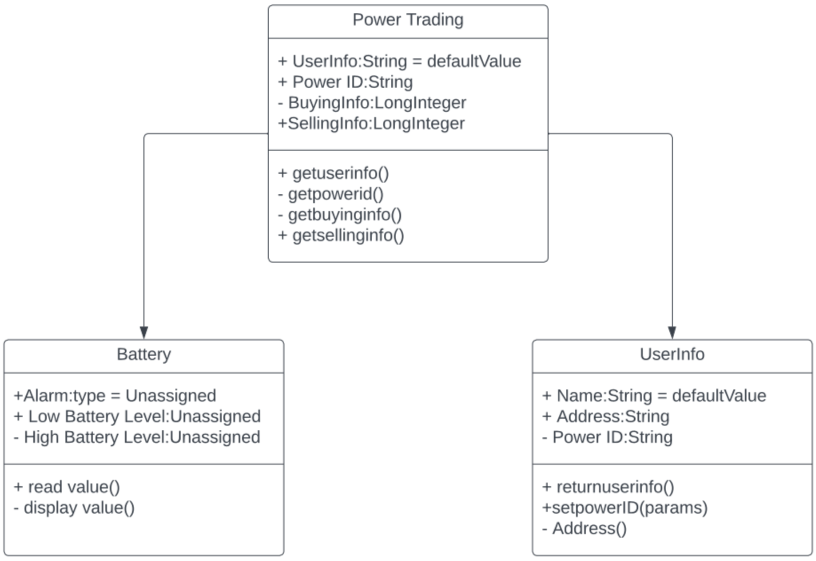
Non-Functional Requirements

- System must have a 99.99% or higher uptime rate

6 use case diagrams

- User must be able to interact with the system on a casual basis
- System must be able to bear multiple alternate sources of energy

Object Class Diagram

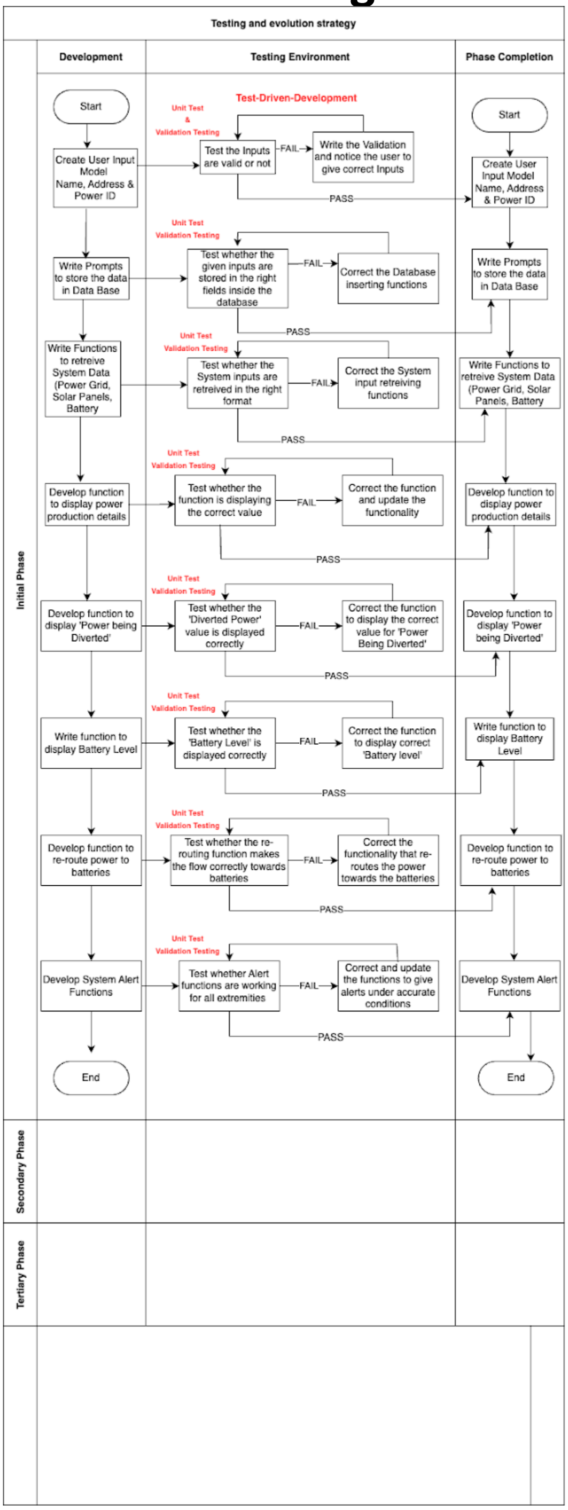


Test and Evolution Planning

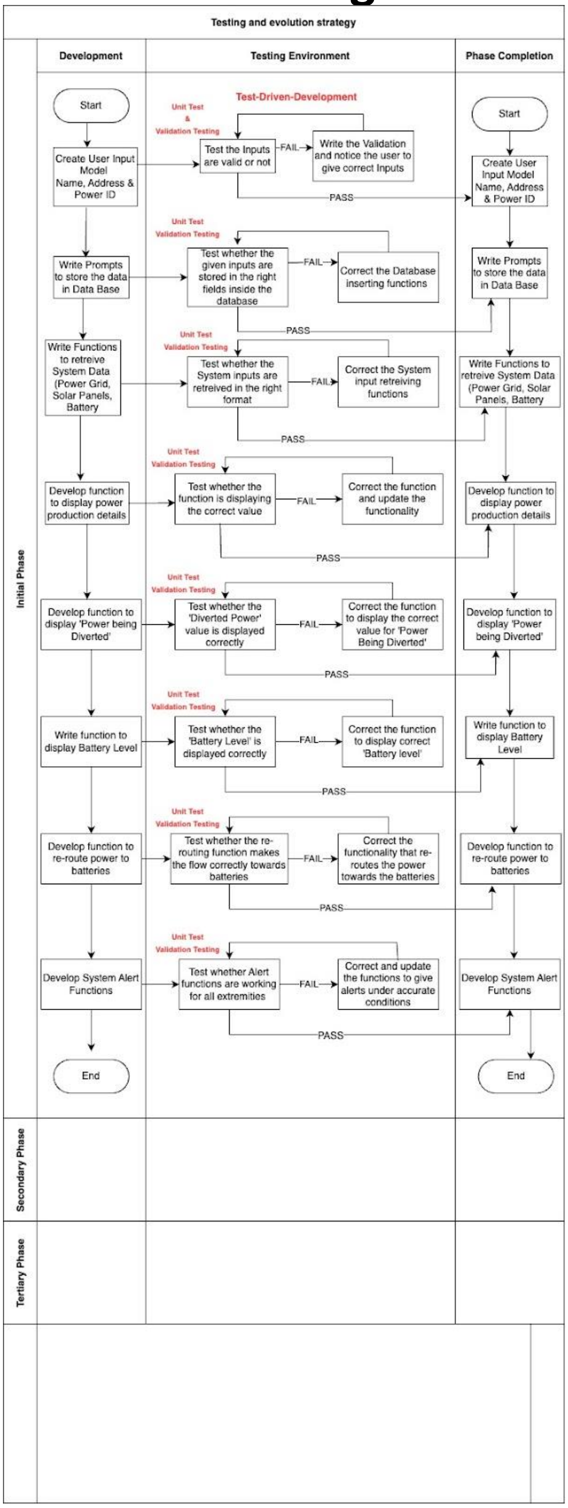
Initial Phase:

Initial phase includes testing during the development as our model adapts TDD (Test-Driven Development) in which unit testing and validation testing is done initially along with the development of features. If the phase passes the test, then it is marked as completed and goes on to next tests.

6 use case diagrams



6 use case diagrams



6 use case diagrams

Secondary Phase:

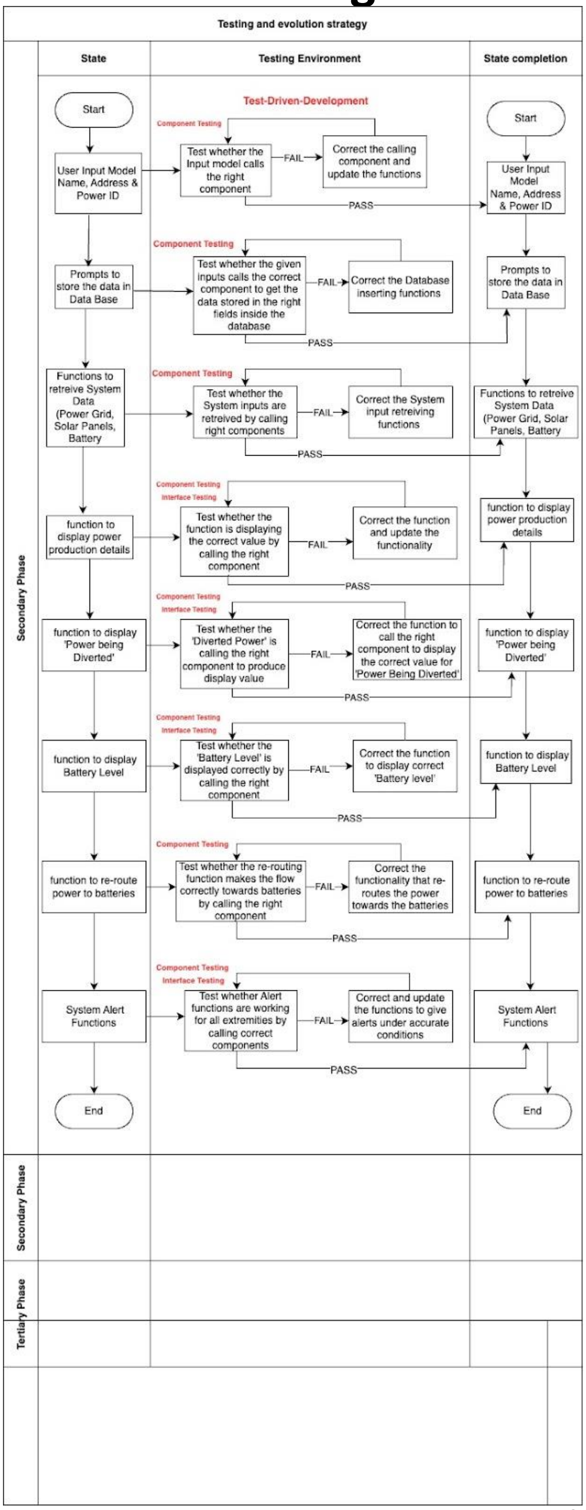
Secondary Phase includes Component testing and Interface testing

Component testing covers these errors:

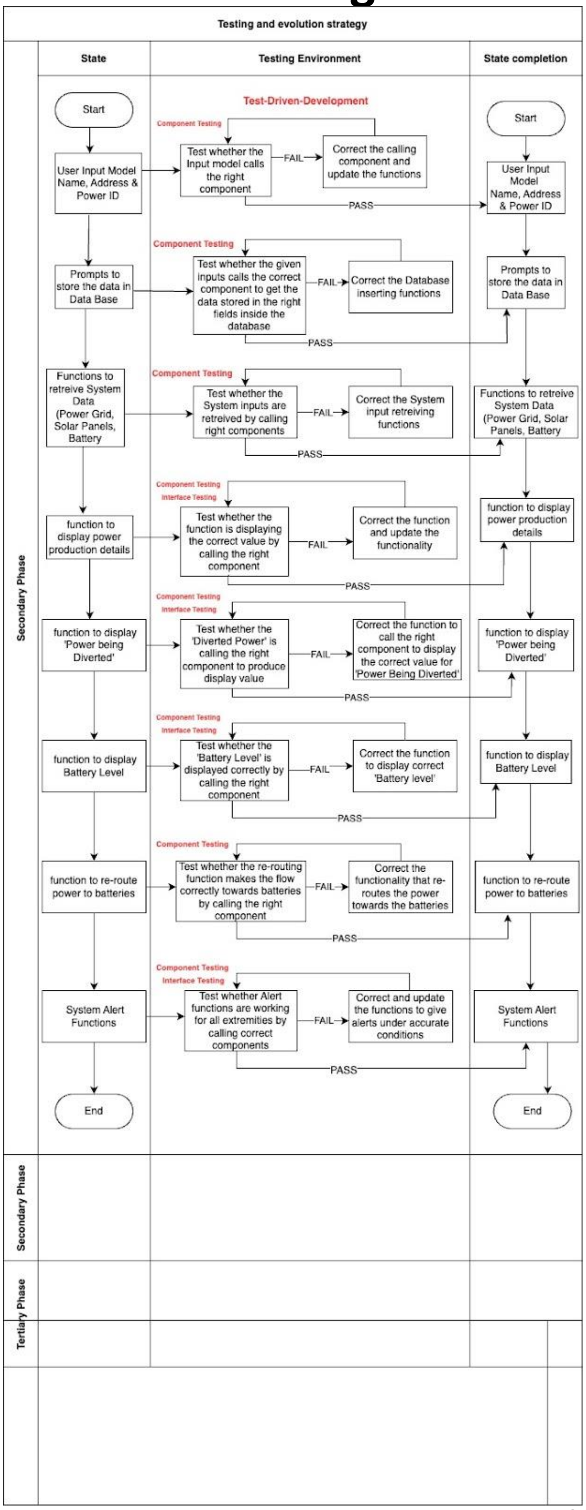
- Interface Misuse
- Interface Misunderstanding
- Timing Errors

Interface testing is done because it ensures that an end-user does not face or encounter any major or minor hindrance while using a particular software or application.

6 use case diagrams



6 use case diagrams



6 use case diagrams

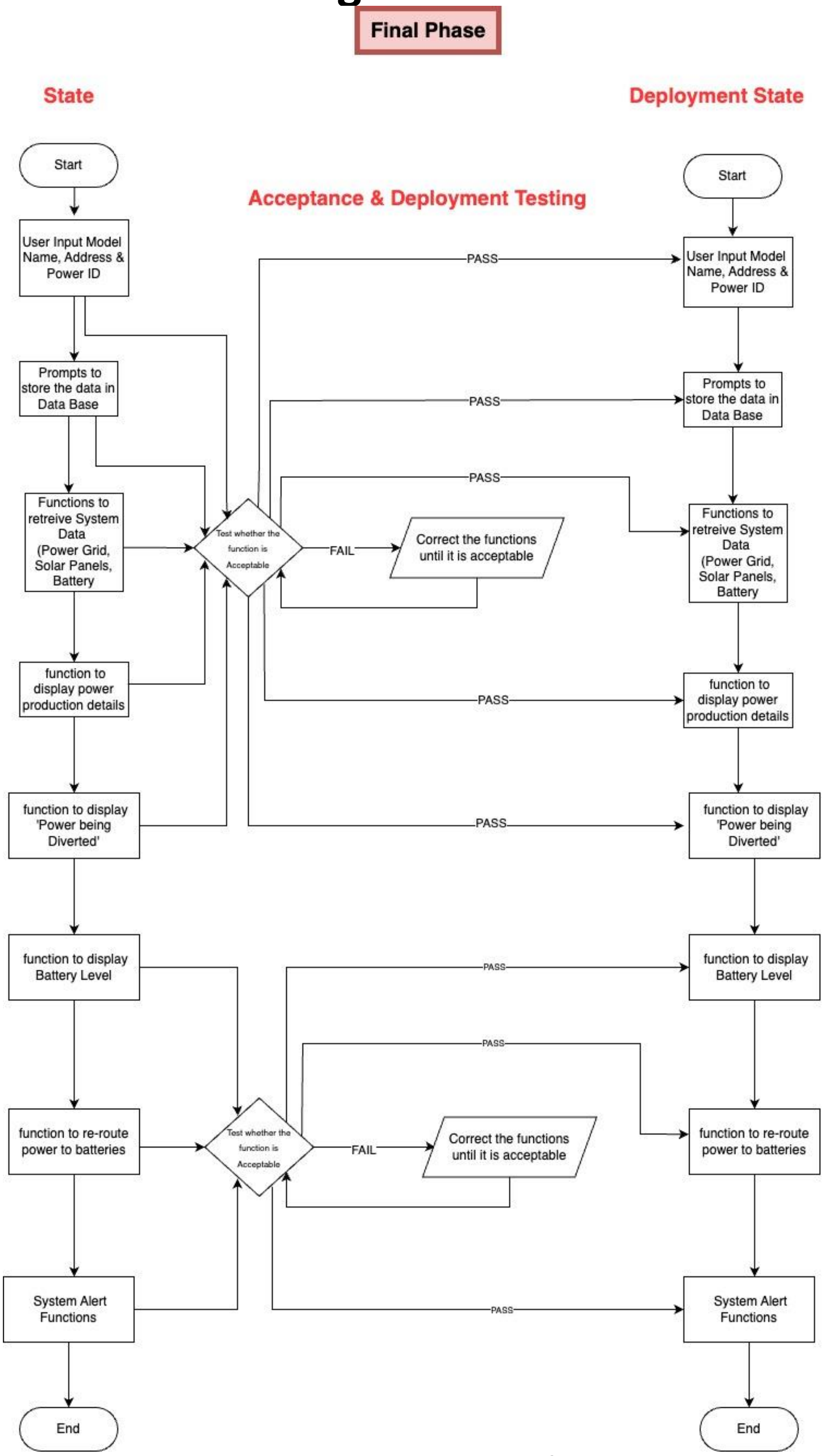
Final Phase:

Final phase of testing includes acceptance testing in which every functionality is tested for acceptance by the developers and the user. If the functionality passes the test, then it is forwarded to deployment testing and when it passes that level, the functionality is deployed. If the function does not pass the deployment testing, then it’s reverted back to the acceptance testing phase.

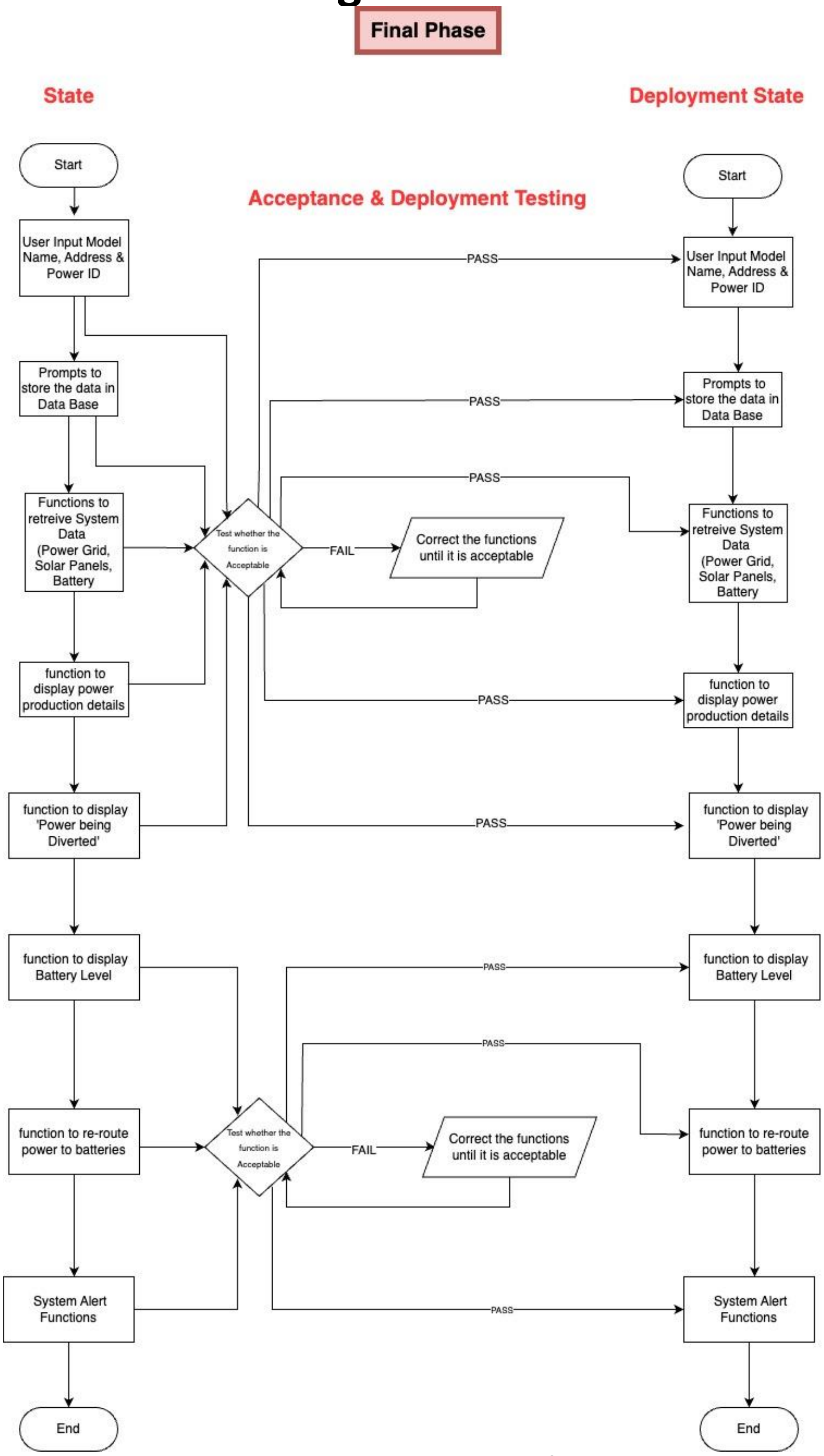
Acceptance tests cover the following attributes:

- User Stories
- Acceptance Criteria
- Use Cases

6 use case diagrams



6 use case diagrams



6 use case diagrams

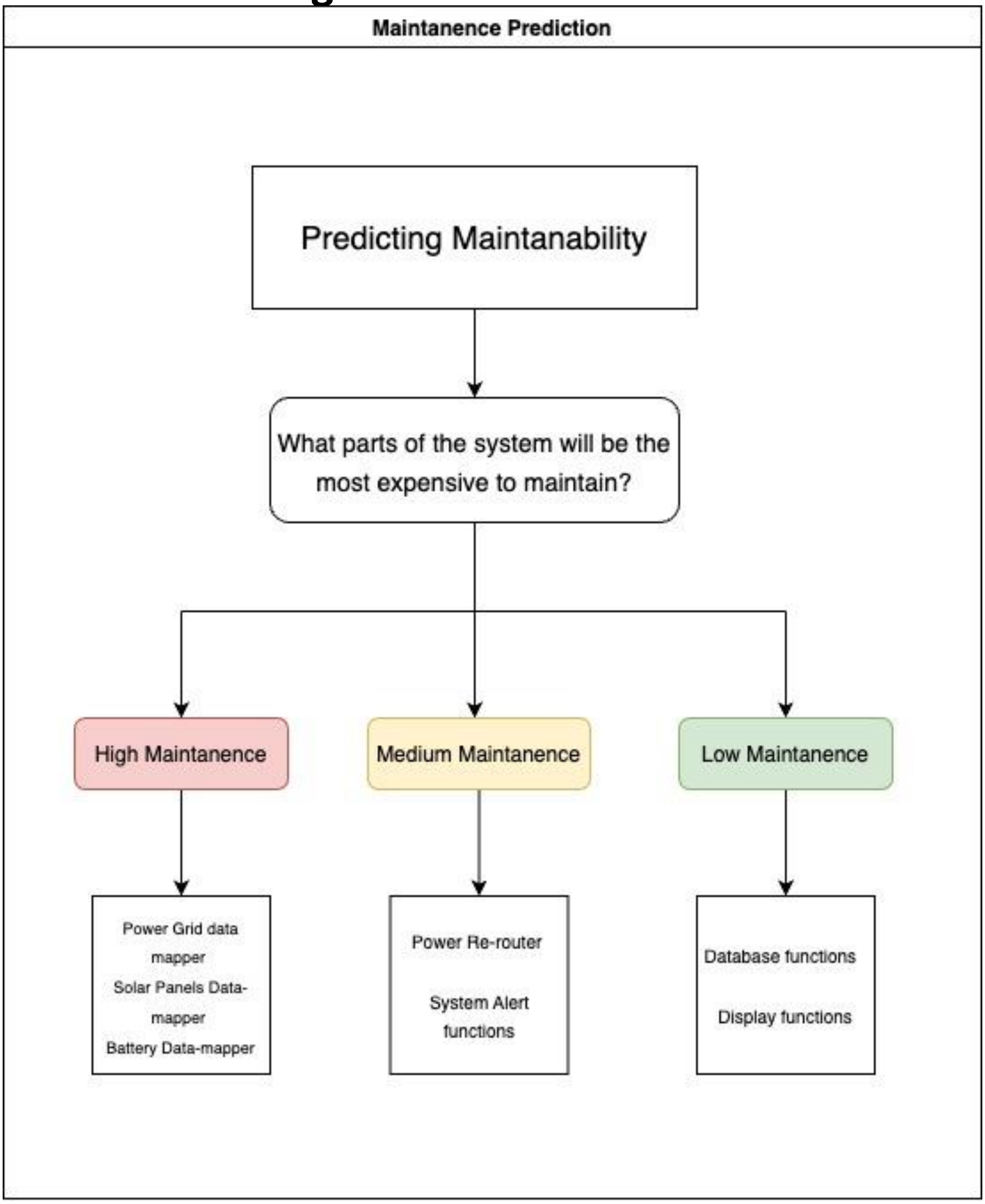
Maintenance Plan:

Maintenance plan includes three main attributes. They are predicting maintainability, Predicting System Changes and Budget Plan

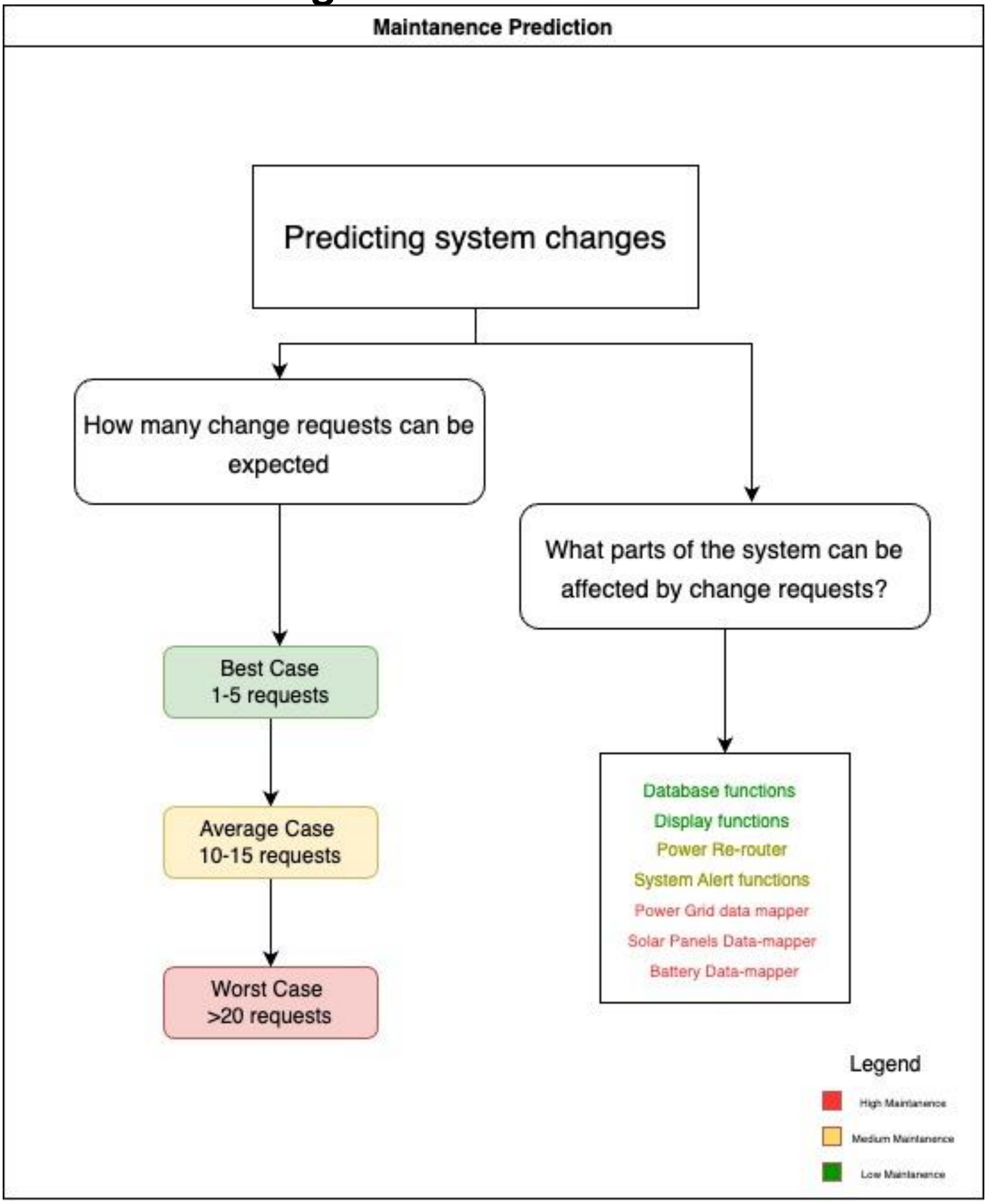
Predicting Maintainability & System Changes:

Maintainability prediction validates which part of the system will be most expensive to maintain, here we categorized those parts into High, medium and low priority components. Similarly System changes are sorted according to requests

6 use case diagrams



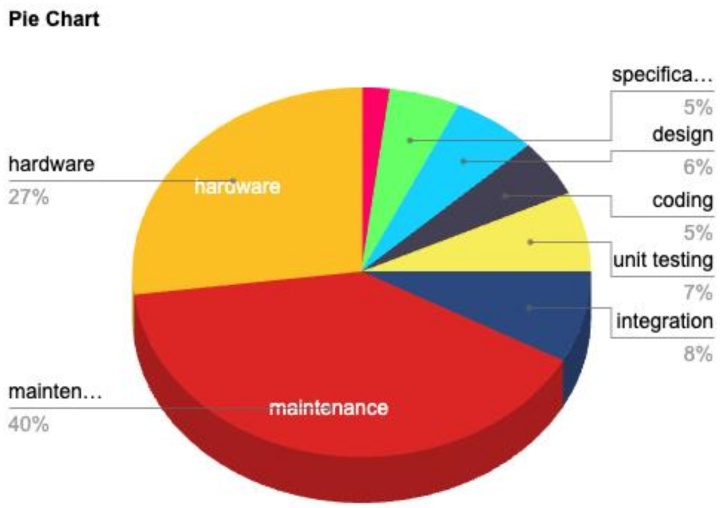
6 use case diagrams



6 use case diagrams

Budget

Breakdown:



Measurement Plan

Planning
Trello
i. Enables the use of Kanban Principles.
ii. Visual representation of work done.

6 use case diagrams

iii. Measures task throughput.
iv. Measures low throughput areas.
Organization
Jira
v. Enables the use of Agile Principles
vi. Assignment of development tasks
vii. Work monitoring
viii. Measures team efficiency
Version Control
GitHub
ix. Version control
x. Version branching
xi. Individual member contribution
xii. Measures issue tracking and resolution
xiii. Measures high tracked areas
xiv. Measures improvement needs

Software Reliability:

The likelihood that software will operate faultlessly for a predetermined amount of time in a predetermined environment is known as software reliability. System reliability is significantly influenced by software reliability as well. In contrast to hardware reliability, it emphasizes design excellence rather than manufacturing excellence by calculating in accordance with the number of assets present in the project.

Assets in our project:

- Database input prompts
- System data retriever
- Solar-Panels data mapper
- Power grid Data mapper
- Power production data display
- Power Diverted data display
- Battery level data display
- Power reroute
- System Alert functions

Total Number of Assets ~ 9

Let's have the **hours of operation as 5,000** for example

Downtime depends on the following factors in our project:

Human error: Human mistake is one of the main reasons for unplanned downtime, whether it is unintentional or the result of carelessness. Costly downtime might result from an user/admin accidentally unplugging a cable, accidentally deleting data, or not adhering to regular procedures. (Ex: entering wrong inputs for power data)

Hardware / software failure: Application failure and system outages are more likely to occur when hardware or software is outdated. Ineffective performance from outdated technology and software also has a negative impact on productivity. (Ex: power re-router has stopped working)

Device misconfiguration: Misconfigured devices are a significant contributor to unplanned downtime. Your network may become vulnerable to cyberattacks due to configuration mistakes that lead to security weaknesses. Instead of manually setting the parameters, you can automate the process to prevent configuration problems. (Ex: Battery level data breach by external attacks)

Bugs: Operating system bugs in servers can affect their performance as well as create security problems. Patches can corrupt programs and cause server failure if they aren't applied on schedule or without the proper testing. (Ex: faults in power calculation algorithm)

Cybersecurity threats: One of the most serious and frequent sources of IT downtime, cyberthreats, including sophisticated ransomware and phishing assaults, can put your business at a stop. Malicious actors can quickly take advantage of weaknesses in your network to infiltrate systems, access sensitive data, and more.

Natural disasters: Natural calamities like hurricanes, floods, and earthquakes can interfere with communication and the electrical grid and even harm electronics. If the downtime lasts for a long time, it could have disastrous effects on our product.

While down-time is unavoidable, it can be monitored and prevented using following actions

6 use case diagrams

- Developing a disaster recovery plan
- Ensuring our hardware / software devices are up-to-date
- Testing all the backups regularly
- Constantly monitoring our data resources, network and associated devices
- Training the users with latest updates

Software reliability heavily depends on the calculation of software reliability metrics

Software reliability Metrics:

- Mean Time to Failure (MTTF)
- Mean Time to Repair (MTTR)
- Rate of occurrence of failure (ROCOF)
- Mean Time Between Failure (MTBF)
- Probability Of Failure On Demand (POFOD)
- Availability (AVAIL)

Mean Time to Failure (MTTF):

MTTF looks at how much time has elapsed between two failure occurrences, and it's averaged over the total number of failures

Formula of **MTTF** = Total hours of operation / Total assets in use

MTTF in our project:

MTTF = 5,000 / 9

MTTF = 555.55 hours

We can conclude that the average lifespan of our assets is around 555 hours

Mean Time To Repair (MTTR):

MTTR is the average time that it takes to repair something after a failure

Formula of **MTTR** = Total down Time / number of breakdowns

MTTR in our project:

As total down time can not be predicted, as we discussed earlier it depends on several factors, but for example, on a average, if a system breaks down for 5 times per 1000 hours and let's say the downtime is around 1 hour, it gives **25 breakdowns for 5,000 hours** and the **down time is 25 hours**, then MTTR can be calculated as follows**.**

MTTR = 25 breakdowns / 25 hours

MTTR = 1 hour

We can conclude that the average time taken to recover is 1 hour

Rate of occurrence of Failure (ROCOF)

ROCOF (rate of occurrence of failures) is the probability that a failure occurs in a given time interval.

6 use case diagrams

Formula of **ROCOF** = Number of Items Failed / Total time taken

ROCOF in our project:

As number of items failed can be unpredictable too, let's say there are **5 failures** over the span of **5,000 operating** hours

ROCOF = 5 / 5000

ROCOF = 0.001

We can conclude that rate of occurrence of failure is 0.001

Mean time between Failure (MTBF)

When a component, assembly, or system fails, the MTBF is the amount of time that has passed, assuming a constant failure rate. Simply, it is the inverse of ROCOF

Formula of **MTBF** = 1 / (Number of Items Failed / Total time taken)

MTBF = Total time taken / Number of Items Failed

MTBF in our project:

As number of items failed can be unpredictable too, let's say there are **5 failures** over the span of **5,000 operating** hours

MTBF = 5000 / 5

MTBF = 1,000 hours

We can conclude that the average time between a failure is 1,000 hours

Probability of Failure on Demand (POFOD):

the unreliability of that component or system is referred as the probability of failure on Demand

Formula of **POFOD =** 1 - Dependability

POFOD in our project:

Let's analyze the POFOD our project in three different scenarios

Best Case:

In best case, let's say the dependability of our project is high, **99.9% (0.999)**

POFOD = 1 - 0.999

POFOD = 0.001

Average Case:

In best case, let's say the dependability of our project is high, **95% (0.95)**

POFOD = 1 - 0.95

POFOD = 0.05

Worst Case:

In best case, let's say the dependability of our project is high, **90% (0.90)**

POFOD = 1 - 0.90

POFOD = 0.1

Therefore, we can conclude that POFOD can vary with the dependability of our project which also varies with several use cases we defined earlier

6 use case diagrams

Availability (AVAIL) :

The probability that a system, at a point in time, will be operational and able to deliver the requested services

Calculating Availability:

Availability, as a measure of uptime, can be calculated as follows:

Percentage of availability = (total elapsed time – sum of downtime) / total elapsed time

Elapsed time = Actual time taken while particular event is occurring
Total downtime = Total number of minutes in a given calendar month that the site is not available

Calculating Availability assuming no planned downtime for our project

Availability Level	Allowed unavailability window					
	Per Year	Per Quarter	Per Month	Per Week	Per Day	Per Hour
90%	36.5 days	9 days	3 days	16.8 hours	2.4 hours	6 mins
95%	18.25 days	4.5 days	1.5 days	8.4 hours	1.2 hours	3 mins
99%	3.65 days	21.6 hours	7.2 hours	1.68 hours	14.4 mins	36 secs
99.5%	1.83 days	10.8 hours	3.6 hours	50.4 mins	7.20 mins	18 secs
99.9%	8.76 hours	2.16 hours	43.2 mins	10.1 mins	1.44 mins	3.6 secs
99.95%	4.38 hours	1.08 hours	21.6 mins	5.04 mins	43.2 secs	1.8 secs
99.99%	52.6 minutes	12.96 mins	4.32 mins	60.5 secs	8.64 secs	0.36 secs

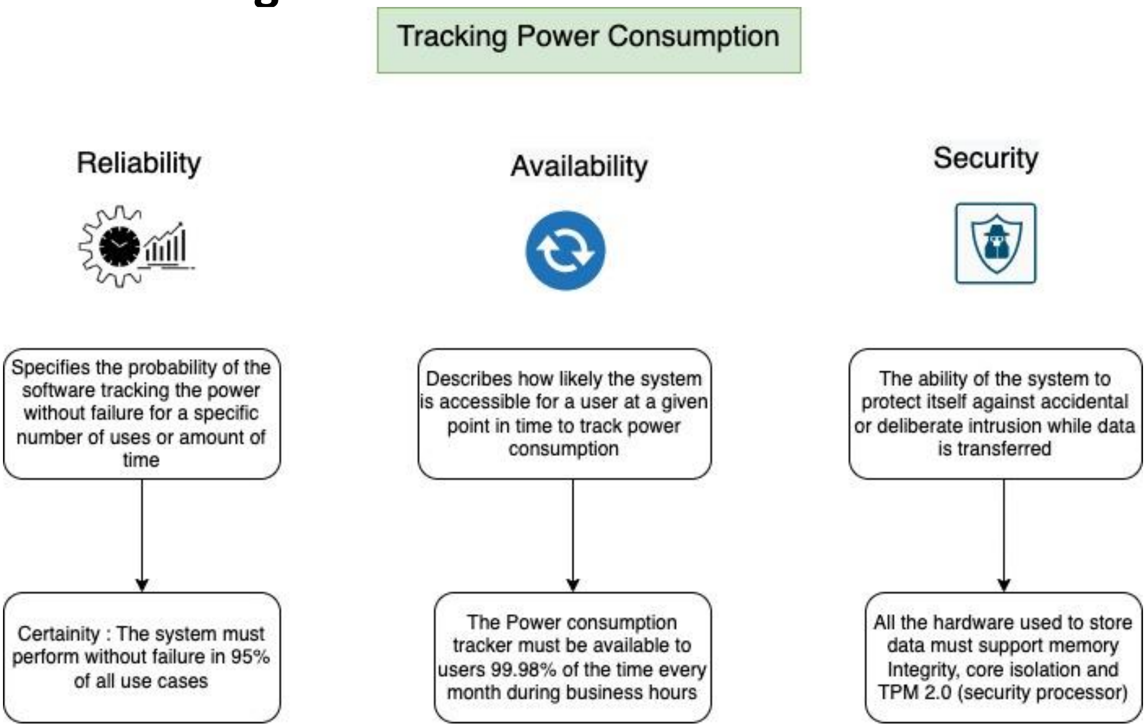
The number of users affected due to an outage and the length of the outage are not taken into account

Safety, Security and Resilience:

Threats to EMSS Reliability:

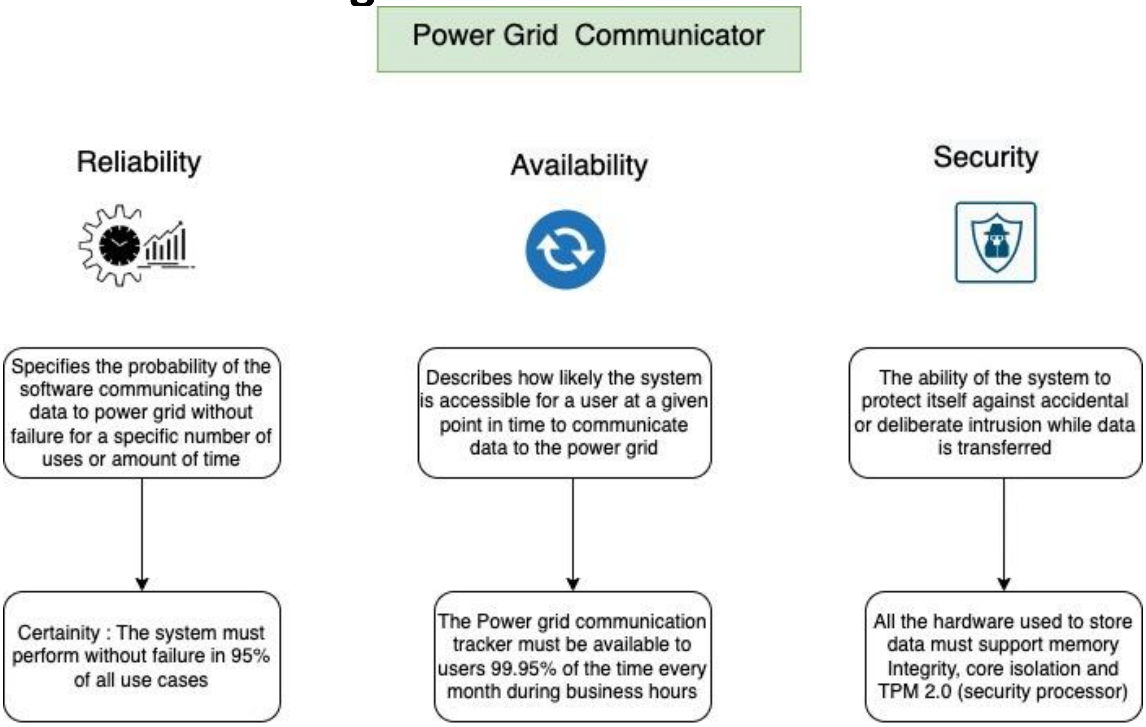
1. During our development process the development should take into consideration certain cases that may affect the long-term operation of the project. For example certain failures in the tracking of power consumption (i.e. a test of software resilience in the form of data corruption by programming error or outside malicious attackers) over time can lead to power generation down the line.

6 use case diagrams



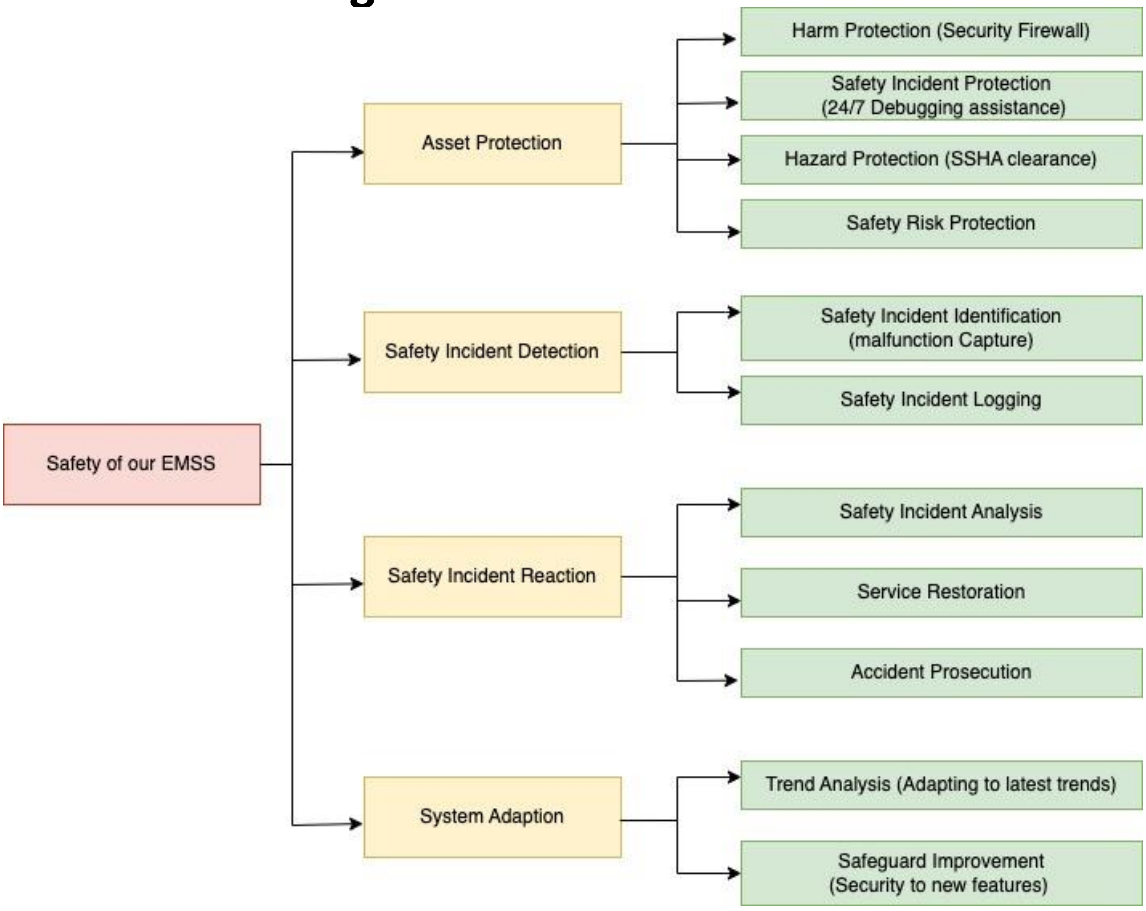
2. Failure through program error to accurately track consumption/selling of power is a major issue. If the software we are running to communicate to the power grid the need to receive or sell off power goes down then we can experience a severe failure in our program.

6 use case diagrams



Safety Configuration of our EMSS:

6 use case diagrams



Mitigating Risk:

1. To Mitigate the Risk to Hardware:

i). If we see a catastrophic environmental event (earthquake damages delicate hardware or ash darkens the sky for extended periods of time limiting the solar power received) we would want our software to implement a redundancy that would see power converting to a limited alternative source (i.e. our secondary generator).

ii). Additionally we should want to try to diversify the channels in which data is stored to our database. A good idea is to send information in time-delayed sequence. For example, we can track the consumption of power on certain preset increments. We can have two functions to simultaneously track the data. One function can send data for energy tracking initially. The second function will 'hold' that data and then after a preset time check to see if the data sent from the first function made it over successfully. If it did not that function will update the data stored. If the data was stored accurately then the function will 'dump' that data and resume tracking.

iii) If our panels get disconnected from the

2. To Mitigate the Risk to Software:

i). If we have a critical error to our software such as stack overflow or pointer corruption

6 use case diagrams

ii). A development could be an error to the update of power flow due to a software bug (power wasn't distributed on the normally scheduled time due to an edge case such as Daylight Savings Time interrupting the regularly timed code). This would cause a system error.