Project Report Format

REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

INRODUCTION :

In the past couple of decades, technology has been at the forefront of every revolutionary change that is occurring at
both global and local levels. These techniques have led to ake the lives of humans much more simplified, smooth,
and even comfortable at different levels. Fro the age of the internet to the age of electric vehicles, technology and
especially engineering has led to inventions and major level improvements on each spectrum imaginable and even
beyond that. Especially in the past decade, the widespread use of the internet along with personal computing devices
has also brought major changes to the lifestyle of the people around the globe. From getting daily news updates to
booking flights to even groceries to our doorstep, the world has come a long way through the use of technology. The
new spectrum under research in technology is artificial intelligence, mostly abbreviated to AI. The world does not
want just machines to do what they are told but even expect devices to work like us. Machine Learning, a subsection
of AI, is the hottest technology right now being implemented o a daily basis and is to be supposed to reach its peak
in the next decade.
Now, as the world has become a better place by providing everything at ease through technology to humans, we
need to utilize the technology for differently-abled people. These people are blind, deaf, dumb, or those who suffer
from cognitive disabilities. The main aim is to make the internet easily accessible to differently-abled people,

PURPOSE:

The technology to be used here would be object recognition, Optical Character Recognition(OCR)[2], text to

speech, speech to text conversions. These could be used in different forms. Object Recognition] is great computer
technology. Humans can easily detect and identify objects present in an image. The human visual system is fast and
accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious
thought [4-10].

Another technology to be used is Optical character recognition. OCR[2][16] (optical character recognition) is the
use of technology to distinguish printed or handwritten text characters inside digital images of physical documents,
such as a scanned paper document. The basic process of OCR involves examining the text of a document and
translating the characters into code that can be used for data processing. OCR is sometimes also referred to as text
recognition.
The combination of OCR and Object detection[1] is of application to the people with disabilities like being blind to
provide the sense of there surrounding and travel without any help or even be safe while traveling, providing
direction[17] through these things combined and provide feedback based whether there is some vehicle or something
in front on them.
These both can also be applied even to classroom learning, providing students with interactive learning for
students[13] who are blind to have a sense of their books and even what is being taught.

LITERATURE SURVEY:

Braille Script was a remarkable invention in the 17th century for the blind; it provided the people with disabilities to

read without the conventional method of eyes rather than through the sense of touch. So this was the perfect
example, as things came per se the books, they were then tweaked for people with disabilities so they can also
benefit from reading at their comfort and through their senses.
The last couple of decades have been remarkable in the revolution through computer science and its technology. The
world has become more tightly bound to each other, even with the distances among them, comfort has increased,
and the lifestyle has become more technology-driven than ever before. These benefits should also reach the
differently-abled people. They should also be able to have a life at their comfort through the use of technology. They
should also be able to use their devices at ease. This is the major motivation for this proje

EXISTING PROBLEM:

Braille Script was a remarkable invention in the 17th century for the blind; it provided the people with disabilities to
read without the conventional method of eyes rather than through the sense of touch. So this was the perfect

AI and automation are two of the key reasons for modernizing. That's why it's vital to know that your choice of AI servers, storage and software can determine how well you are able to infuse AI throughout your organization, reduce the cost of data ingress and egress, and protect your data for a trustworthy AI framework.

REFERENCES:

1] Bigham, J. P., Jayant, C., Miller, A., White, B., & Yeh, T. (2010, June). VizWiz:: LocateIt-enabling blind
people to locate objects in their environment. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops (pp. 65-72). IEEE

. Fusion: Practice and Applications (FPA) Vol. 1, No. 1, PP. 32-39, 2020
DOI: 10.5281/zenodo.3825929 39

[2] Manduchi, R., Kurniawan, S., & Bagherinia, H. (2010, October). Blind guidance using mobile computer vision: A usability study. In Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility (pp. 241-242).

[3] Ivanchenko, V., Coughlan, J., Gerrey, W., & Shen, H. (2008, October). Computer vision-based clear path
guidance for blind wheelchair users. In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (pp. 291-292).

[4] Johnsen, A., Grønli, T. M., & Bygstad, B. (2012). Making touch-based mobile phones accessible for the
visually impaired. Norsk informatikkonferanse,(Bodø, Norway, 201

PROBLEM STATEMENT DEFINITION:

A problem statement is a tool to help guide your team toward building the right product. Defining AI problem statements well is important in pointing a path to a worthwhile AI product.

It's easy to solve the wrong problems. Good design relentlessly questions assumptions, reframing the design problem to be solved. We know this, and yet, HOW to actually reframe a problem is missing from our conversations. - Stephen Anderson

AI problem statements fall into one of two traps.

- **Too broad** to be meaningful, e.g. talking about value to users, adding intelligence, or automating X.
- **Too prescriptive** so that we venture into the 'how to' territory.

It is quite an art to thread the line between open-ended yet focused without drifting toward excessively broad or overly prescriptive. It is an art worth practicing.

But first, some nuance. Stating that a problem is an AI problem is prescriptive in and of itself.

Nevertheless, here are some **rules of thumb that I've found useful in avoiding bad problem statements** when designing products that are likely to include AI.

IDEATION &PROPOSED SOLUTION:

EMPATHY MAP CANVAS:

Robots need love too — Empathy Mapping for AI

Product people build things that serve some purpose and solve some problem. Unfortunately, when it comes to machine learning, most are forgetting this.

We need a way to set guidelines on how the non-deterministic, intelligent algorithms should work. You can’t sprinkle it like magic dust and assume it will be transformative.

At Philosophie, we use prototyping to quickly try more solutions for our clients. This helps them reduce the likelihood of failure when going to market. For machine learning we have started to experiment with building non-coded prototypes that simulate what machine learning could do. Transitioning from prototype to building is not as straightforward. We needed a good way to do this and found inspiration from a research technique…

IDEATION &BRAIN STORMING:

Teams can easily collaborate, solve problems, and generate ideas by using MURAL as a shared online space for design thinking.

MURAL is more than a replacement for your office whiteboard. It is a platform where ideas can come to life, meetings are more interactive, and teams can innovate faster. Teams can use MURAL to shape their ideas when they are not sitting together in the same room. In addition to the visual features, like sticky notes, icons, and diagramming, MURAL has more than 300 templates that enable you to implement and scale design thinking and agile practices.

In addition to design thinking activities, IBM uses MURAL for conducting technical discovery sessions and leading other types of collaborative meetings and workshops.

PROPSED SOLUTION:

Natural Language Processing (NLP) enables understanding, interaction and communication between humans and machines. Our AI solutions use NLP to automatically extract critical business insights and emerging trends from large amounts of structured and unstructured content

PROBLEM SOLUTION FIT:

As an emerging technology, AI can transform the way people work with data and with each other. For unified communications, that potential means **improved access to information and better control of UC applications and endpoints**

REQUIREMENT ANALYSIS:

FUNCTINAL REQUIREMENTS:

Visual impairment: this concerns far-sightedness and near-sightedness so there are two types of visual impairment to distinguish between.

⨂ **Hearing impairment**: you can be affected by hearing loss as soon as you lose 20 decibels. It may affect one or both of your ears. Depending on their hearing loss, hearing impaired people can have hearing aids, cochlear implants, subtitles. When we refer to deaf people, this means they can't hear anymore or barely.

⨂ **Intellectual disability**: the WHO defines it as "a significantly reduced ability to understand new or complex information and to learn and apply new skills (impaired intelligence)".

⨂ **Physical disability**: this includes people with a physical impairment or reduced mobility. Thus, their mobility capacity may be limited in their upper and/or lower body.

Business requirements

These include high-level statements of goals, objectives, and needs. Business requirements do not include any details or specific features. They just state the problem and the business objective to be achieved such as

- increased revenue/throughput/customer reach,
- reduced expenses/errors,
- improved customer service, etc.

User (stakeholder) requirements

The needs of discrete stakeholder groups (top-level managers, nonmanagement staff, customers, etc.) are specified to define what they expect from a particular solution. This group serves as a bridge between the generalized business requirements and specific solution requirements. They are outlined in a User Requirements Specification and can include, for example, ability to create various reports, view order history and status,

NON FUNCTIONAL REQUIREMENTS:

not related to the system functionality, rather define *how* the system should perform. Some examples are:

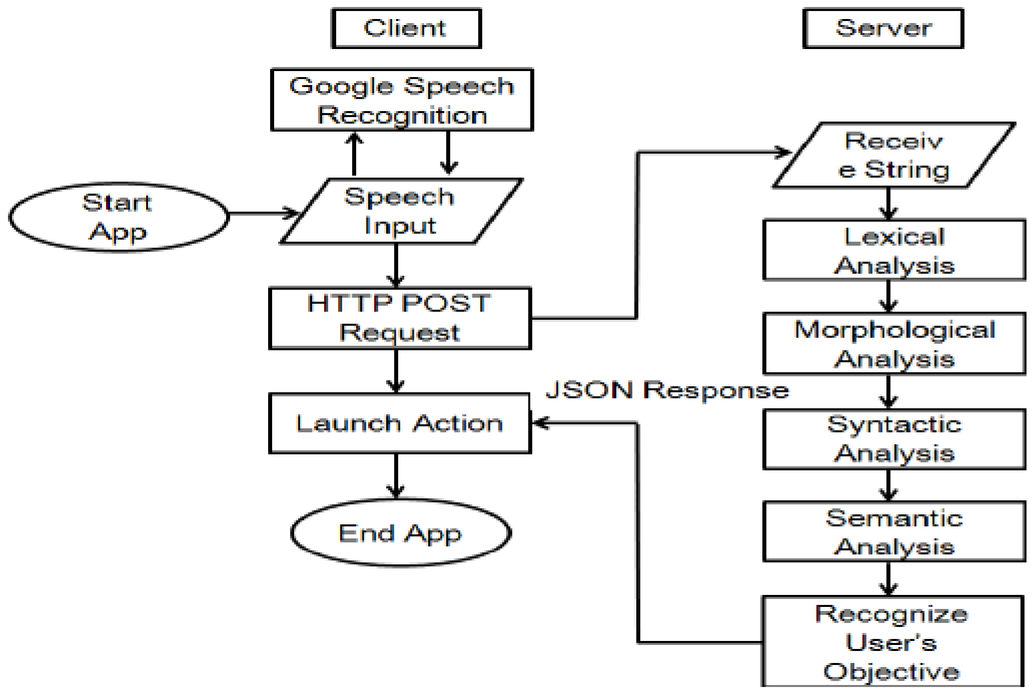
The website pages should load in 3 seconds with the total number of simultaneous users <5 thousand.

The system should be able to handle 20 million users without performance deterioration.

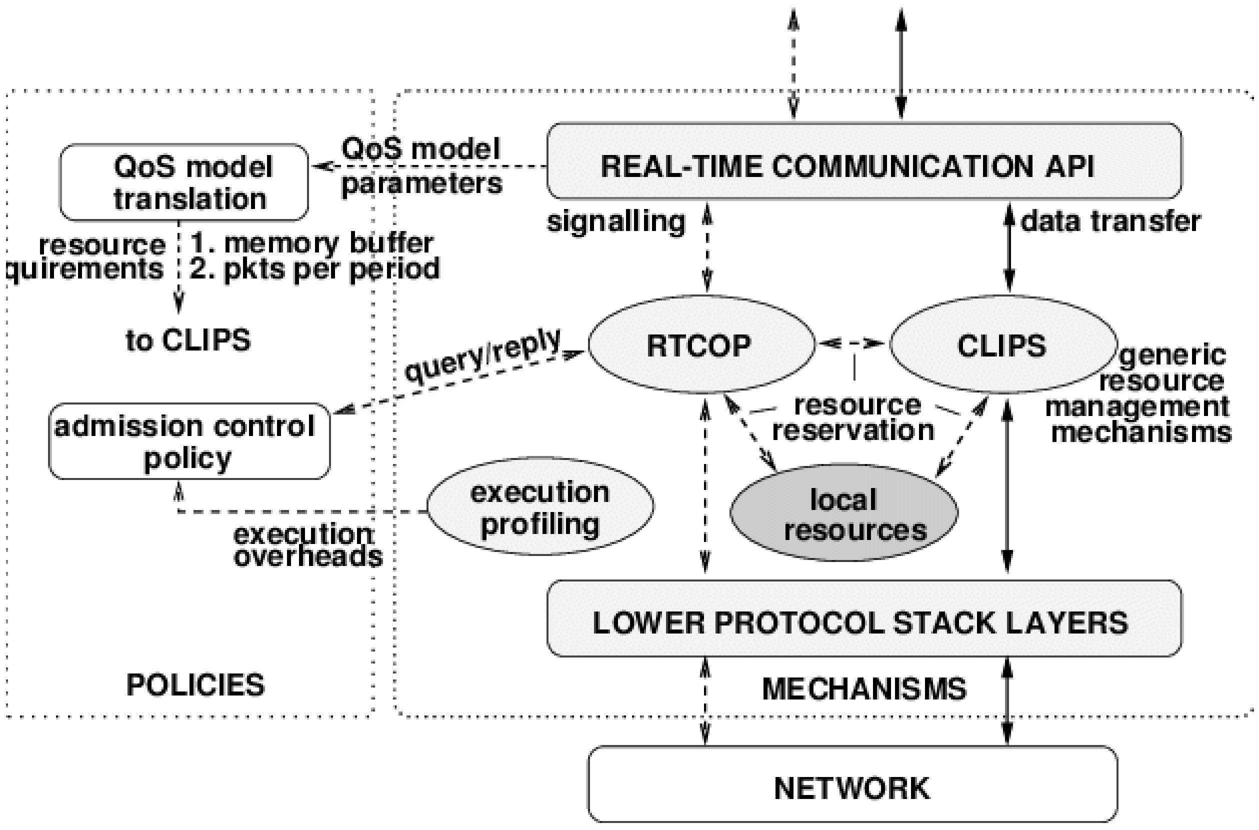
Here's a brief comparison and then we'll proceed to a more in-depth explanation of each group.

PROJECT DESIGN:

DATA FLOW DIAGRAMS:



SOLUTION AND TECHNICAL ARCHITECTURE:



USER STORIES:

In the Garage Method for Cloud and many other agile methods, one of the key tools to communicate between the product owner (the customer) and the development team is the user story. Martin Fowler and Kent Beck define a user story as "...a chunk of functionality (some people use the word *feature*) that is of value to the customer... The shorter the story the better. The story represents a concept and not a detailed specification. *A user story is nothing more than an agreement that the customer and the developers will talk together about a feature*." [1] (Author's emphasis.)

PROJECT PLANNING AND AND SCHEDULING

Feature selection (*sprint planning – part one*)

Many teams set an overall goal for the iteration to help guide the selection of features. At the beginning of the meeting, the highest priority features are typically selected from the release plan. If the iteration does have an overarching goal, then some lower priority features may be selected if they better align with the goal. Prior velocity is critical to enabling the team to schedule a realistic amount of work.

For example, if the team previously planned to get 40 story points worth of product features, but only successfully delivered 30 story points, then 30 story points should be considered the current velocity for the next iteration. Past velocity estimates compared to actual numbers are useful at the iteration level, the feature level, and the task level. All of these help the team determine how much they can sign up for in the next iteration. If the iteration is overbooked, then the customer needs to select which features need to be delayed to a future iteration. During the iteration planning meeting, the customer will discuss features with the team and attempt to answer any questions the team has.

Task planning (*sprint planning – part two*)

The team will break the features down into tasks. Developers then sign up for tasks and estimate them. Tasks typically range in size from four hours to two days, with most tasks capable of being delivered within a day. Tasks larger than two days should generally be broken down into smaller tasks. Occasionally during task planning a feature is determined to be have been woefully underestimated in the original release plan. In this case, the team will need to work with the customer on providing a corrected estimate and determining what feature or features may need to be delayed as a result.

Iteration adjustments

During the iteration, if there is remaining time after all features have been delivered, then the team can request that the customer identify additional feature(s) to add to the iteration. If, on the other hand, it is obvious that not all features can be delivered, then the team works with the customer to determine which features could be delayed or perhaps split in order to deliver the most value by the iteration deadline.

SPRINT DELIVERY SCHEDULE:

the first step in the sprint planning process. During this step, the team selects the amount of work it can deliver from the prioritized backlog, and using historical velocity as a guide, determines how much to schedule in each sprint.

You can schedule a sprint from the **Sprint Scheduling** page. From this page, you can view sprint details and schedules for individual teams, schedules across all teams for a selected project, including the following summary information:

- Sprint start and end dates
- Cumulative estimate for each sprint (and progress bars showing amount of work completed for active sprints)
- Number of stories and defects to be worked during the sprint with their estimates

Closed Sprints and their detaStay on track of sprint goals and improve retrospectives with data scrum teams can put to use sprint over sprint.

Sprint report

Determine overcommittment and excessive scope creep and understand completed work in each sprint.

Burndown chart

Track progress towards sprint goals to manage progress and respond accordingly.

Release burndown

Track and monitor the projected release date for versions and take action if work is falling behind projected schedule.

Velocity chart

Track work from sprint to sprint to helps teams determine the velocity and better estimate the work a team realistically achieve in future sprints.

Optimize kanban flow for continuous delivery

Better predict future performance and spot bottlenecks with agile reports for kanban teams.

Cumulative flow diagram

Easily spot blockages by seeing the number of issues that increase in any given state.

Control chart

Determine future performance with cycle and lead times for your product, version, or sprint.

Work management made easier with Jira reports

Identify trends and work smarter, with out-of-the-box reports for issue analysis and forecasting in Jira Software.

Issue analysis

Average Age Report

Created vs Resolved Issues Report

Pie Chart Report

Recently Created Issues Report

Resolution Time Report

Single Level Group By Report

Time Since Issues Report

Forecast & management

Time Tracking Report

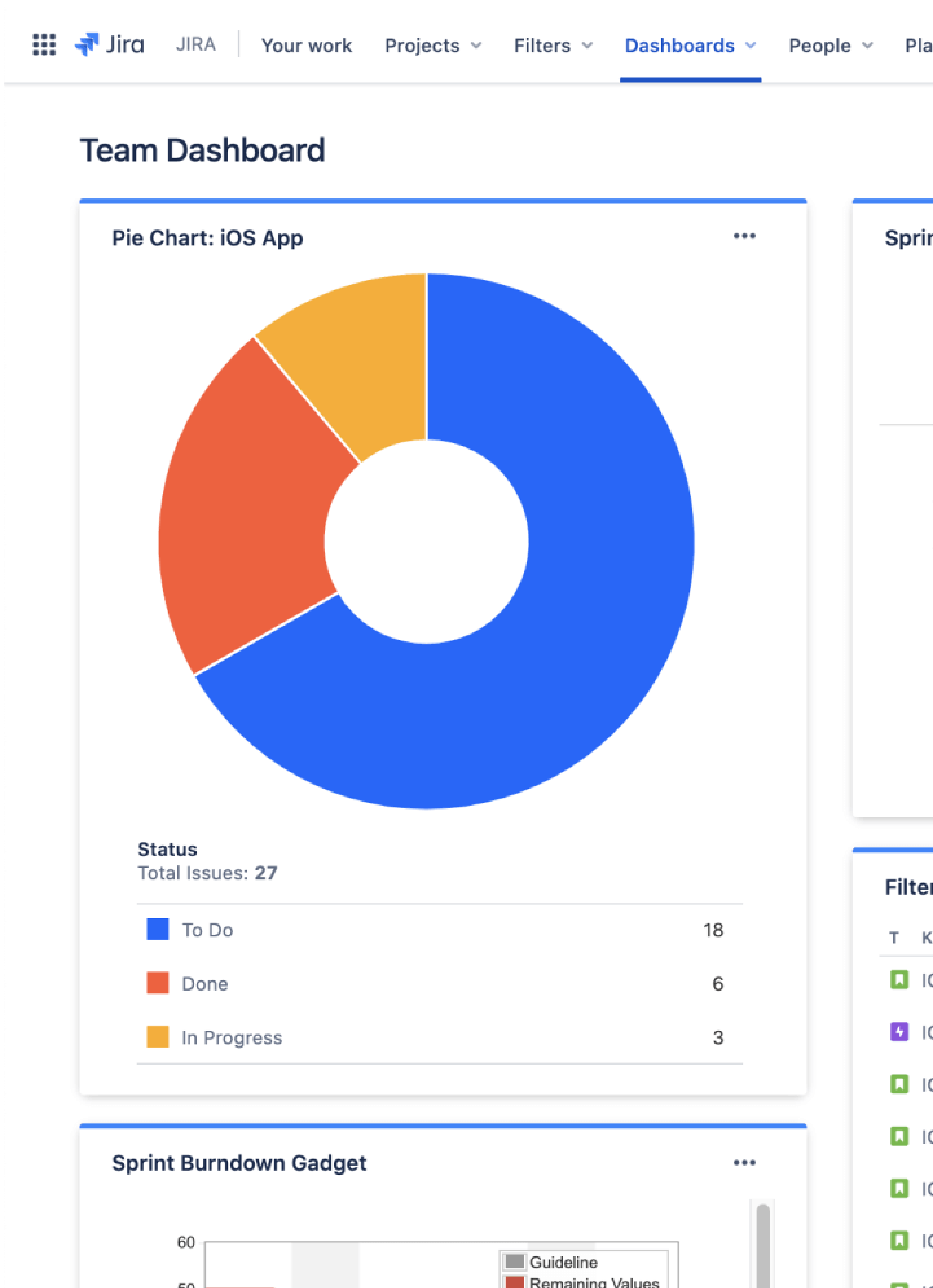
User Workload Report

Version Workload Report

Bring it all together with dashboards in Jira

Organize projects and track achievements in a single view with dashboards in Jira Software. With dozens of built-in gadgets, easily customize dashboards for teams, stakeholders, and leadership.

Learn more



Team Dashboard

Pie Chart: iOS App

Status
Total Issues: **27**

🟦	To Do	18
🟥	Done	6
🟨	In Progress	3

Sprint Burndown Gadget

ils are viewable by selecting "Show Closed".

***Note**: Only the most recent six closed sprints is displayed in the Summary list. You can view all the closed sprints from the Project Details page.

CODING AND SOLUTIONING:

FEATURE1:

A method for solving problems

This method is from the book *How to Solve It* by George Pólya. It originally came out in 1945 and has sold over one million copies.

His problem-solving method has been used and taught by many programmers, from computer science professors (see Udacity's Intro to CS course taught by professor David Evans) to modern web development teachers like Colt Steele.

Let's walk through solving a simple coding problem using the four-step problem-solving method. This allows us to see the method in action as we learn it. We'll use JavaScript as our language of choice. Here's the problem:

Create a function that adds together two numbers and returns that value.

There are four steps to the problem-solving method:

1. Understand the problem.
2. Devise a plan.
3. Carry out the plan.
4. Look back.

FEATURE 2:

Step 1: Understand the problem.

When given a coding problem in an interview, it's tempting to rush into coding. This is hard to avoid, especially if you have a time limit.

However, try to resist this urge. Make sure you actually understand the problem before you get started with solving it.

Read through the problem. If you're in an interview, you could read through the problem out loud if that helps you slow down.

As you read through the problem, clarify any part of it you do not understand. If you're in an interview, you can do this by asking your interviewer questions about the problem description. If you're on your own, think through and/or Google parts of the question you might not understand.

This first step is vital as we often don't take the time to fully understand the problem. When you don't fully understand the problem, you'll have a much harder time solving it.

To help you better understand the problem, ask yourself:

What are the inputs?

What kinds of inputs will go into this problem? In this example, the inputs are the arguments that our function will take.

Just from reading the problem description so far, we know that the inputs will be numbers. But to be more specific about what the inputs will be, we can ask:

Will the inputs always be just two numbers? What should happen if our function receives as input *three* numbers?

Here we could ask the interviewer for clarification, or look at the problem description further.

The coding problem might have a note saying, "You should only ever expect two inputs into the function." If so, you know how to proceed. You can get more specific, as you'll likely realize that you need to ask more questions on what kinds of inputs you might be receiving.

Will the inputs always be numbers? What should our function do if we receive the inputs "a" and "b"? Clarify whether or not our function will always take in numbers.

Optionally, you could write down possible inputs in a code comment to get a sense of what they'll look like:

```
//inputs: 2, 4
```

Next, ask:

What are the outputs?

What will this function return? In this case, the output will be one number that is the result of the two number inputs. Make sure you understand what your outputs will be.

Create some examples.

Once you have a grasp of the problem and know the possible inputs and outputs, you can start working on some concrete examples.

Examples can also be used as sanity checks to test your eventual problem. Most code challenge editors that you'll work in (whether it's in an interview or just using a site like Codewars or HackerRank) have examples or test cases already written for you. Even so, writing out your own examples can help you cement your understanding of the problem.

Start with a simple example or two of possible inputs and outputs. Let's return to our addition function.

Let's call our function "add."

What's an example input? Example input might be:

```
// add(2, 3)
```

What is the output to this? To write the example output, we can write:

```
// add(2, 3) ---> 5
```

This indicates that our function will take in an input of 2 and 3 and return 5 as its output.

Create complex examples.

By walking through more complex examples, you can take the time to look for edge cases you might need to account for.

For example, what should we do if our inputs are strings instead of numbers? What if we have as input two strings, for example, add('a', 'b')?

Your interviewer might possibly tell you to return an error message if there are any inputs that are not numbers. If so, you can add a code comment to handle this case if it helps you remember you need to do this.

```
// return error if inputs are not numbers.
```

Your interviewer might also tell you to assume that your inputs will always be numbers, in which case you don't need to write any extra code to handle this particular input edge case.

If you don't have an interviewer and you're just solving this problem, the problem might say what happens when you enter invalid inputs.

For example, some problems will say, "If there are zero inputs, return undefined." For cases like this, you can optionally write a comment.

```
// check if there are no inputs.

// If no inputs, return undefined.
```

For our purposes, we'll assume that our inputs will always be numbers. But generally, it's good to think about edge cases.

Computer science professor Evans says to write what developers call *defensive* code. Think about what could go wrong and how your code could defend against possible errors.

Before we move on to step 2, let's summarize step 1, understand the problem:

```
-Read through the problem.

-What are the inputs?

-What are the outputs?

Create simple examples, then create more complex ones.
```

2. Devise a plan for solving the problem.

Next, devise a plan for how you’ll solve the problem. As you devise a plan, write it out in pseudocode.

Pseudocode is a plain language description of the steps in an algorithm. In other words, your pseudocode is your step-by-step plan for how to solve the problem.

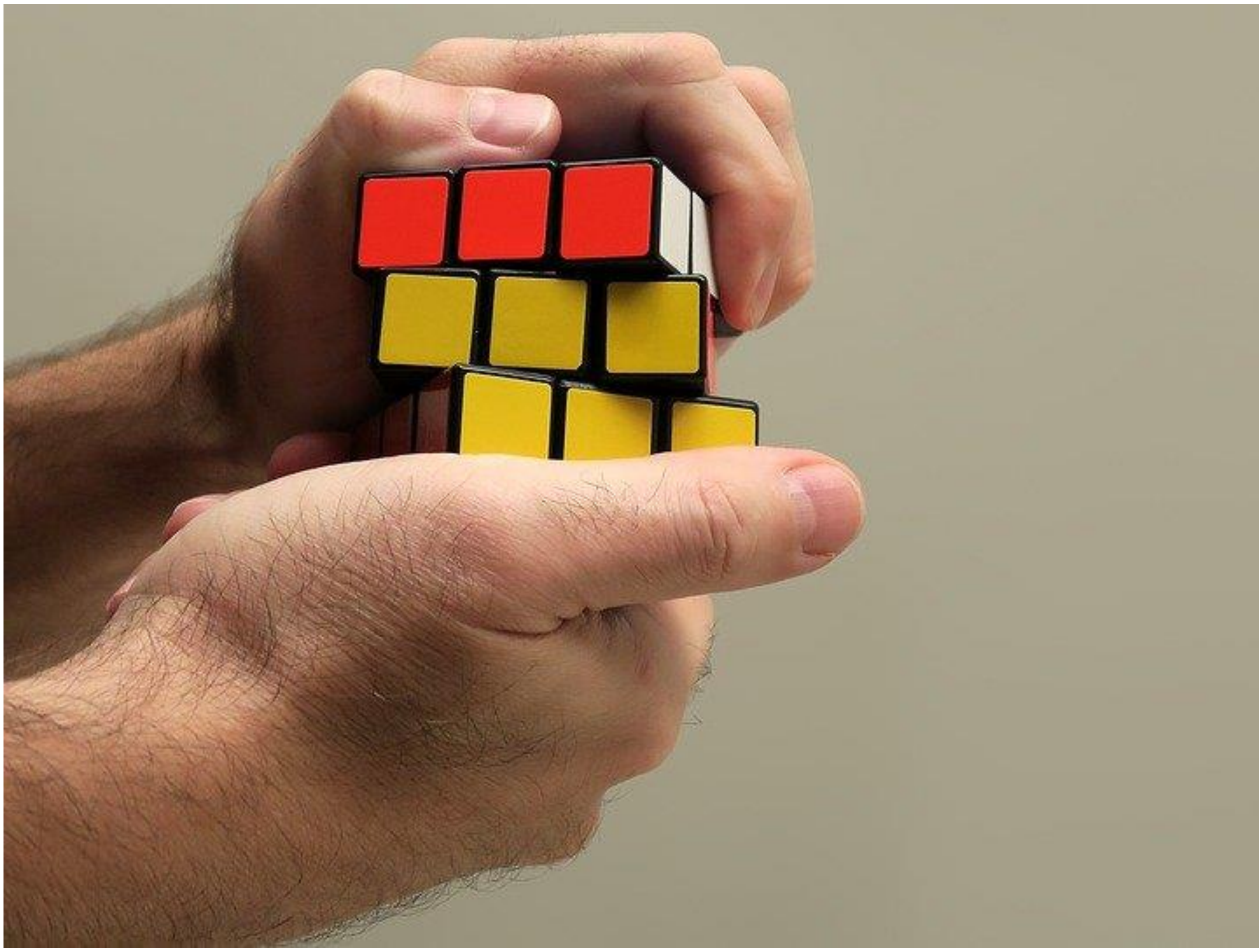
Write out the steps you need to take to solve the problem. For a more complicated problem, you’d have more steps. For this problem, you could write:

```
// Create a sum variable.

Add the first input to the second input using the addition operator.

// Store value of both inputs into sum variable.

// Return as output the sum variable.
```

Now you have your step-by-step plan to solve the problem.

For more complex problems, professor Evans notes, “Consider systematically how a human solves the problem.” That is, forget about how your code might solve the problem for a moment, and think about how *you* would solve it as a human. This can help you see the steps more clearly.

3. Carry out the plan (Solve the problem!)



The next step in the problem-solving strategy is to solve the problem. Using your pseudocode as your guide, write out your actual code.

Professor Evans suggests focusing on a simple, mechanical solution. The easier and simpler your solution is, the more likely you can program it correctly.

Taking our pseudocode, we could now write this:

```
function add(a, b) {
 const sum = a + b;
 return sum;
}
```

Professor Evans adds, remember not to *prematurely optimize*. That is, you might be tempted to start saying, “Wait, I’m doing this and it’s going to be inefficient code!”

First, just get out your simple, mechanical solution.

What if you can’t solve the entire problem? What if there's a part of it you still don't know how to solve?

Colt Steele gives great advice here: If you can’t solve part of the problem, ignore that hard part that’s tripping you up. Instead, focus on everything else that you can start writing.

Temporarily ignore that difficult part of the problem you don’t quite understand and write out the other parts. Once this is done, come back to the harder part.

This allows you to get at least *some* of the problem finished. And often, you’ll realize how to tackle that harder part of the problem once you come back to it.

Step 4: Look back over what you've done.

Once your solution is working, take the time to reflect on it and figure out how to make improvements. This might be the time you refactor your solution into a more efficient one.

As you look at your work, here are some questions Colt Steele suggests you ask yourself to figure out how you can improve your solution:

- Can you derive the result differently? What other approaches are there that are viable?
- Can you understand it at a glance? Does it make sense?
- Can you use the result or method for some other problem?
- Can you improve the performance of your solution?
- Can you think of other ways to refactor?
- How have other people solved this problem?

One way we might refactor our problem to make our code more concise: removing our variable and using an implicit return:

```
function add(a, b) {
 return a + b;
}
```

With step 4, your problem might never feel finished. Even great developers still write code that they later look at and want to change. These are guiding questions that can help you.

If you still have time in an interview, you can go through this step and make your solution better. If you are coding on your own, take the time to go over these steps.

When I’m practicing coding on my own, I almost always look at the solutions out there that are more elegant or effective than what I’ve come up with.

Wrapping Up

In this post, we’ve gone over the four-step problem-solving strategy for solving coding problems.

Let's review them here:

- Step 1: **understand the problem.**
- Step 2: **create a step-by-step plan for how you'll solve it**.
- Step 3: **carry out the plan** and write the actual code.
- Step 4: **look back** and possibly refactor your solution if it could be better.

Practicing this problem-solving method has immensely helped me in my technical interviews and in my job as a developer.

DATABASE SCHEMA:

The Aito database schema is a description of how the database is constructed and internally processed. A schema contains the information of:

- The name of the tables
- The name and the ColumnType of the columns in each table
- The Analyzer of a column if needed
- The relationships (links) between tables

The Aito database requires a defined schema before executing other operations. The schema is defined in the JSON format and populate to Aito using the Schema API Endpoint.

Schema structure

- A ***schema*** defines tables with unique name, mapping from table name to table definition.
- A ***table*** must define:
  - its name
  - "type": "table" as only "table" is currently supported
  - "columns": defines columns with unique name, mapping from column name to column definition
- A ***column*** must define:
  - its name
  - "type": the appropriate column type
  - "analyzer" if needed
  - "link": if needed

An example of the Aito schema:

```
TESTING:
      TESTING CASES:
```

Test case design refers to a structured and sequential list of action items that attempt to verify a specific feature. At the heart of a test case design is a sequence of steps describing actions to be performed, the test data to be used, and an expected response for each action performed.

You should write tests to cover the business, functional, and technical requirements. For adequate test coverage, you can refer to the requirements artifacts, whether they're written in the form of user stories or technical design documents. The test case template and level of detail required will vary depending on the organization, type of software delivery project, and or the test management tool used. In this guide, we'll walk you through the basics of writing test cases and how to write test cases for AI-based software testing tools.

The Objective of Writing Test Cases

- Tests should help assess the specific functionality of a software application.
- Tests should document a sequence of steps to be executed, which can be useful to reference when a bug is found in the application
- To identify issues in user experience and to identify gaps in design at the early stages.

RESULTS:

PERFORMANCE MATRICES:

As a small business owner, you will always want to keep an eye on how your business is performing.

Whether that is keeping an eye on your sales, your customer satisfaction, or even your warehouse efficiency.

A business that is performing well, is a business that is making money.

But how do you measure performance? And what are performance metrics?

Let's take a closer look.

Performance metrics are data used to track processes within a business.

This is achieved using activities, employee behavior, and productivity as key metrics.

These metrics are then used by employers to evaluate performance.

This is in relation to an established goal such as employee productivity or sales objectives.

ADVANTAGES AND DISADVANTAGES:

Artificial Intelligence is one of the emerging technologies which tries to simulate human reasoning in AI systems. **John McCarthy** invented the term Artificial Intelligence in the year 1950.

He said, '**Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions, and concepts, solve kinds of problems now reserved for humans, and improve themselves.**'

Artificial Intelligence is the ability of a computer program to learn and think. Everything can be considered Artificial intelligence if it involves a program doing something that we would normally think would rely on the intelligence of a human.

The advantages of Artificial intelligence applications are enormous and can revolutionize any professional sector. Let's see some of them

1) Reduction in Human Error:

The phrase "**human error**" was born because humans make mistakes from time to time. Computers, however, do not make these mistakes if they are programmed properly. With Artificial intelligence, the decisions are taken from the previously gathered information applying a certain set of algorithms. So errors are reduced and the chance of reaching accuracy with a greater degree of precision is a possibility.

Example: In Weather Forecasting using AI they have reduced the majority of human error.

2) Takes risks instead of Humans:

This is one of the biggest advantages of Artificial intelligence. We can overcome many risky limitations of humans by developing an AI Robot which in turn can do the risky things for us. Let it be going to mars, defuse a bomb, explore the deepest parts of oceans, mining for coal and oil, it can be used effectively in any kind of natural or man-made disasters.

Example: Have you heard about the **Chernoby**l nuclear power plant explosion in Ukraine? At that time there were no AI-powered robots that can help us to minimize the effect of radiation by controlling the fire in early stages, as any human went close to the core was dead in a matter of minutes. They eventually poured sand and boron from helicopters from a mere distance.

AI Robots can be used in such situations where intervention can be hazardous.

3) Available 24x7:

An Average human will work for 4–6 hours a day excluding the breaks. Humans are built in such a way to get some time out for refreshing themselves and get ready for a new day of work and they even have weekly offed to stay intact with their work-life and personal life. But using AI we can make machines work 24x7 without any breaks and they don't even get bored, unlike humans.

Example: Education

CONCLUSION:

rtificial intelligence has the potential to transform all organizations. The process by which this transformation happens can vary, but the steps will tend to follow the roadmap we have listed in this book. Following all the steps outlined in the previous chapters will enable your organization to implement and excel in the use of AI technology. AI holds the key to unlocking a magnificent future where, driven by data and computers that understand our world, we will all make more informed decisions. These computers of the future will understand not just *how* to turn on the switches but *why* the switches need to be turned on. Even further, they may one day ask us if we need *switches* at all.

Although AI cannot solve all your organization's problems, it has the potential to completely change how business is done. It affects every sector, from manufacturing to finance, bringing about never before seen increases in efficiency. As more industries adopt and start experimenting with this technology, newer applications will be invented. AI will bring a change even more widespread and sweeping than the introduction of computing devices. It will change the way we transact, get diagnosed, perform surgeries, and drive our cars. It is already changing industrial processes, medical imaging, financial modeling, and computer vision. We are well on our way to tapping into this enormous potential, and as a result, the future holds better decision-making potential and faster, ..

FUTURE SCOPE:

The feature scope refers to the extent to which a feature is applied. For example, in a multi-body part, we can apply a feature such as an extruded cut and specify which body can be included in the cut and which body should not be included. In our exercise, notice in the drawing provided that there is a hole that goes through bodies 2 and 4 and skips body 3.

To utilize the feature scope, we can follow the same steps as for an extruded boss. However, we will notice options under the Feature Scope tile in our cut extrude PropertyManager. We can see options highlighted in the screenshot with both the sketch and the other options for the extruded cut feature. Under the Feature Scope options, we can select the Selected ...

APPENDICES:

SOURCE CODE:

The feature scope refers to the extent to which a feature is applied. For example, in a multi-body part, we can apply a feature such as an extruded cut and specify which body can be included in the cut and which body should not be included. In our exercise, notice in the drawing provided that there is a hole that goes through bodies 2 and 4 and skips body 3.

To utilize the feature scope, we can follow the same steps as for an extruded boss. However, we will notice options under the Feature Scope tile in our cut extrude PropertyManager. We can see options highlighted in the screenshot with both the sketch and the other options for the extruded cut feature. Under the Feature Scope options, we can select the Selected ...

GIT HUB&PROJECT DEMO LINK:

#

demo-website

Here are 27 public repositories matching this topic...

hwalsuklee / awesome-deep-vision-web-demo

  -

- Code
-

- Issues

-

  - Pull requests

A curated list of awesome deep vision web demo

demo list awesome deep-neural-networks computer-vision deep-learning vision awesome-list demo-website deep-vision vision-demo ai-demo deeplearning-demo deep-learning-demo