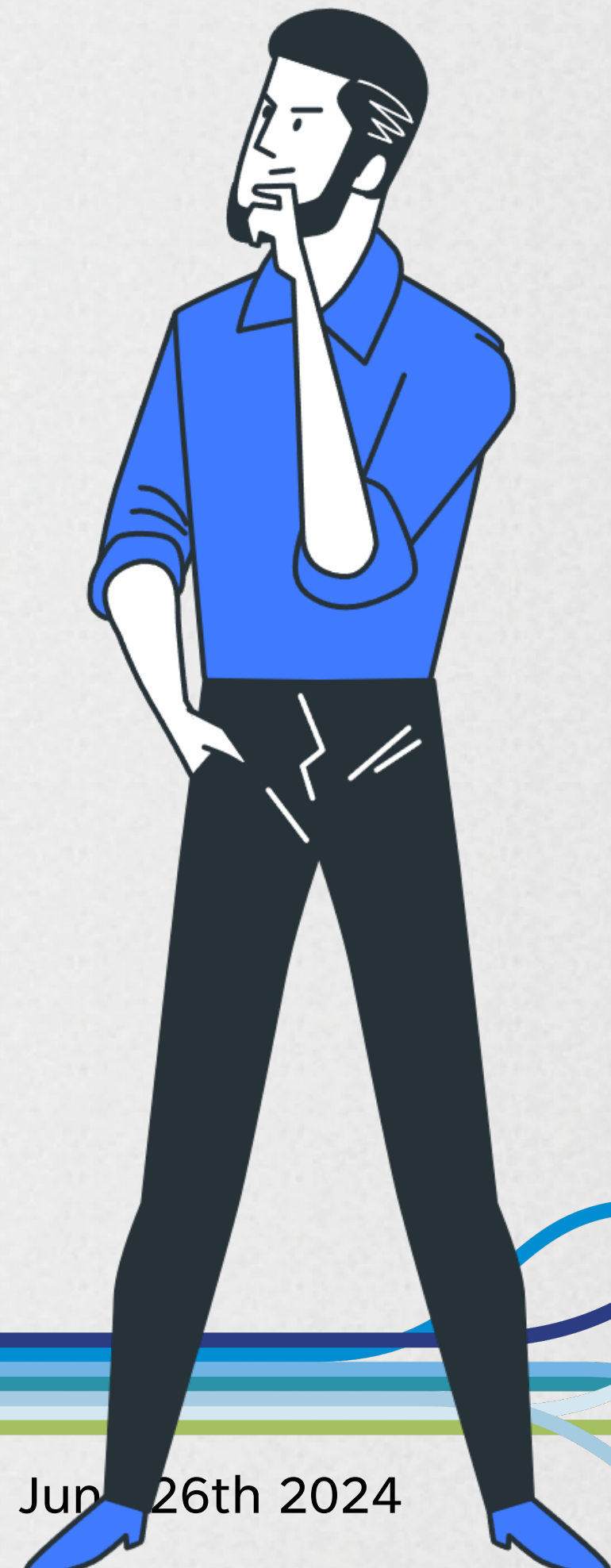




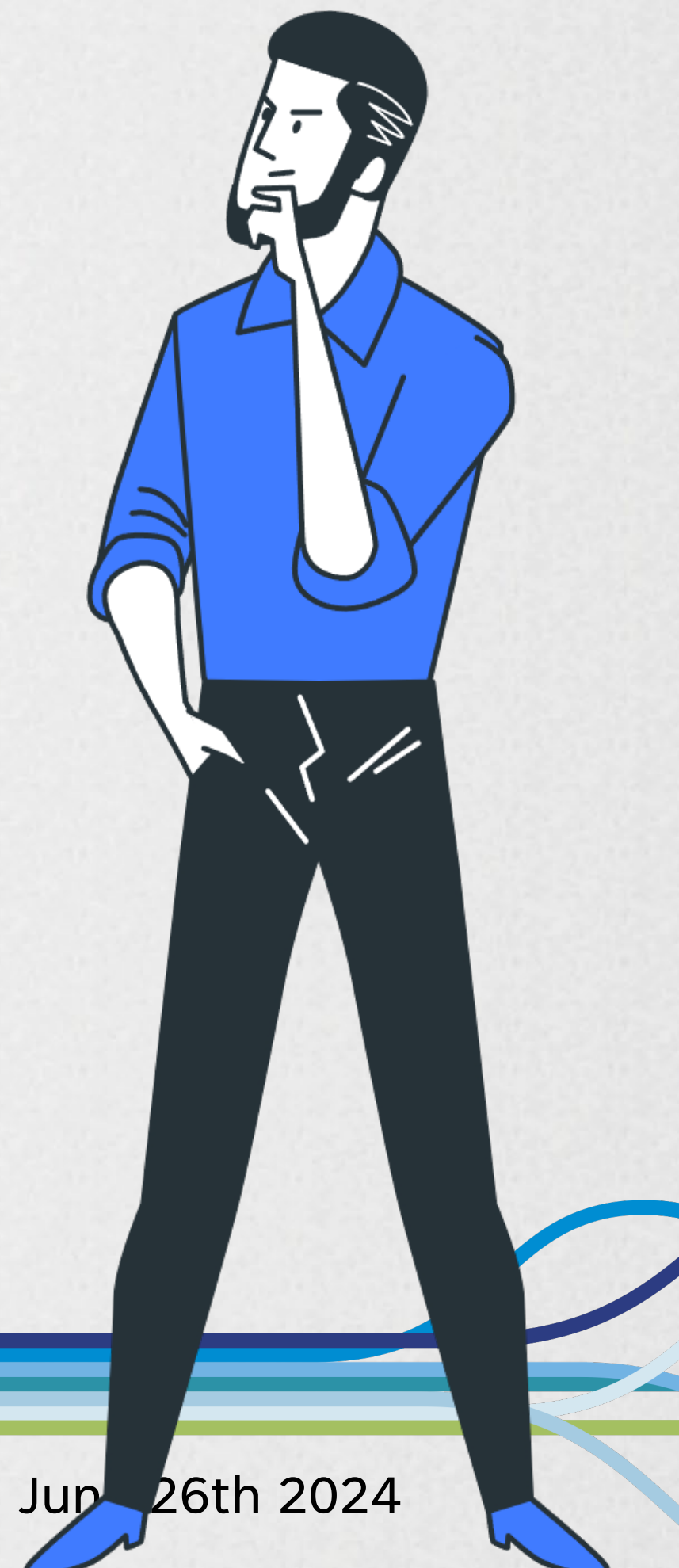
L.C.GPT.f.e.T.C.C. & B.E.i.S.T.





L.C.GPT.f.e.T.C.C. & B.E.i.S.T.

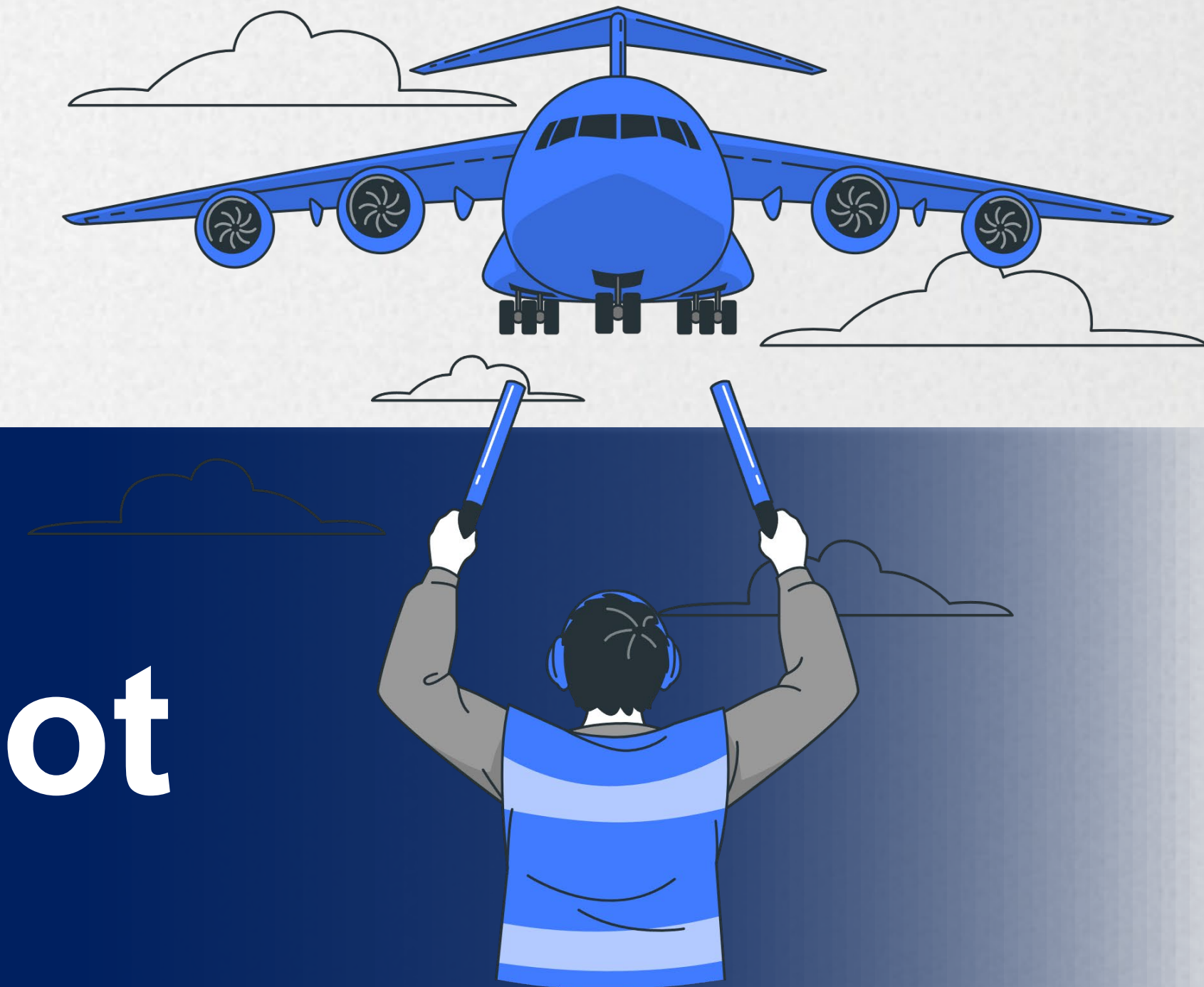
**Leveraging Chat GPT
for efficient Test Case Creation
& Business Engagement in Software Testing**





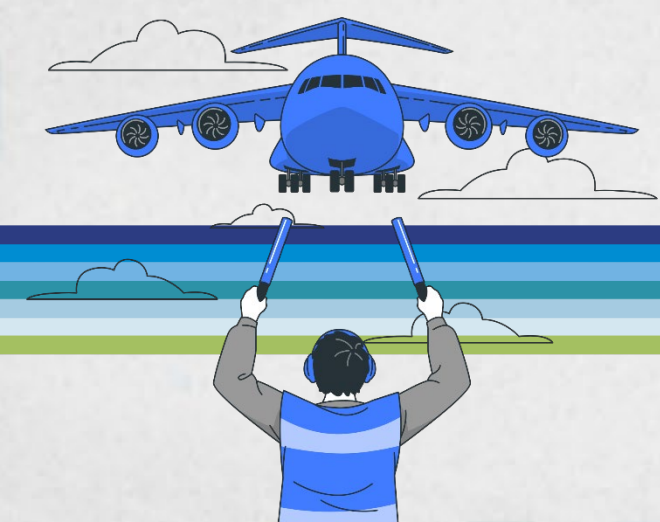
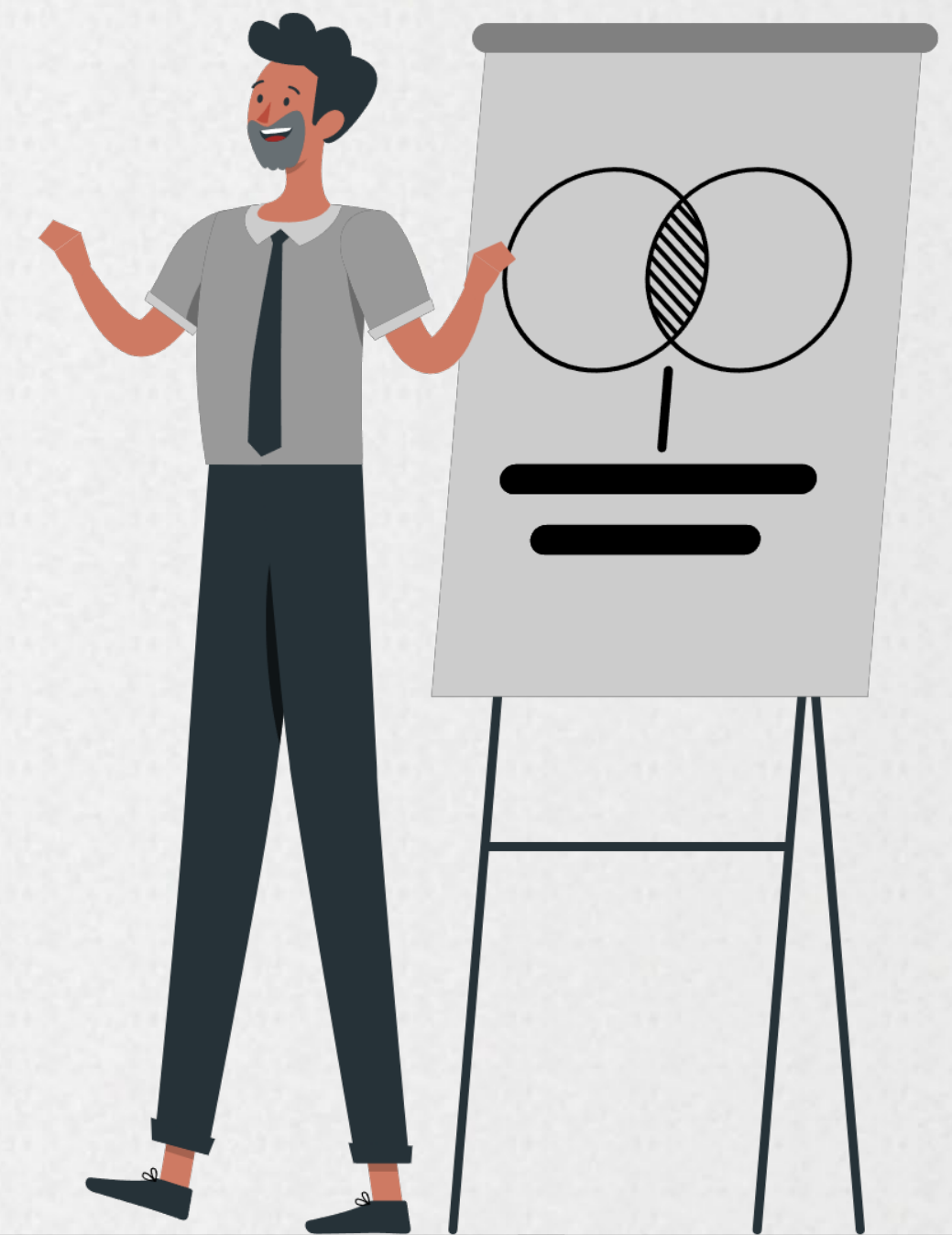
L.C.GPT.f.e.T.C.C. & B.E.i.S.T.

Leveraging Chat GPT
for efficient Test Case Creation
& Business Engagement in Software Testing



Co-Pilot, Not Auto-Pilot

- Introduction
- What we won't cover
- The Rise of Chat GPT
- Challenges in Software Testing
- Examples





Introduction

Who am I

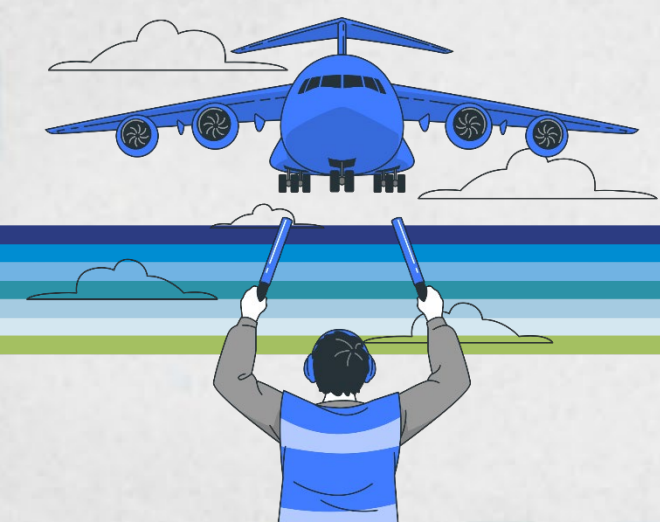
- *I know that I know nothing...*

Why this topic?

- Work smart / not hard!

When and how can you ask questions?

- Always 😊





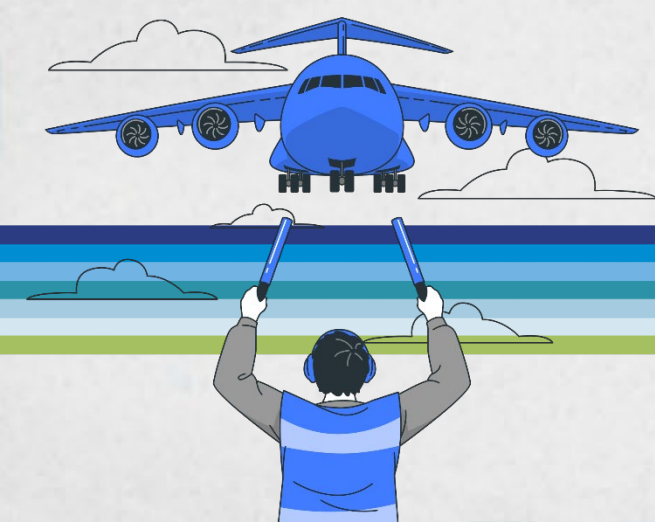
Won't cover...

Not enough time...

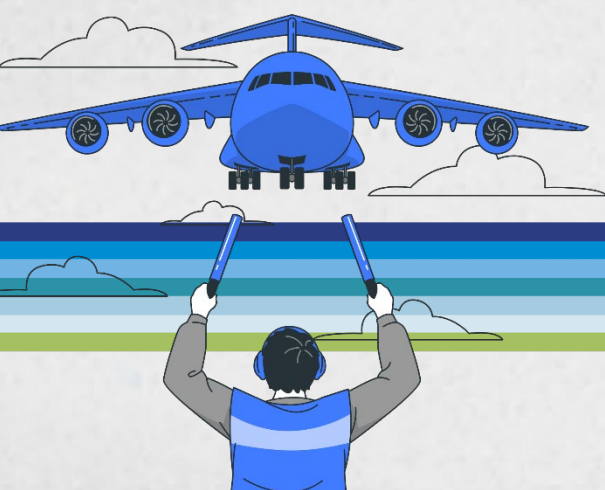
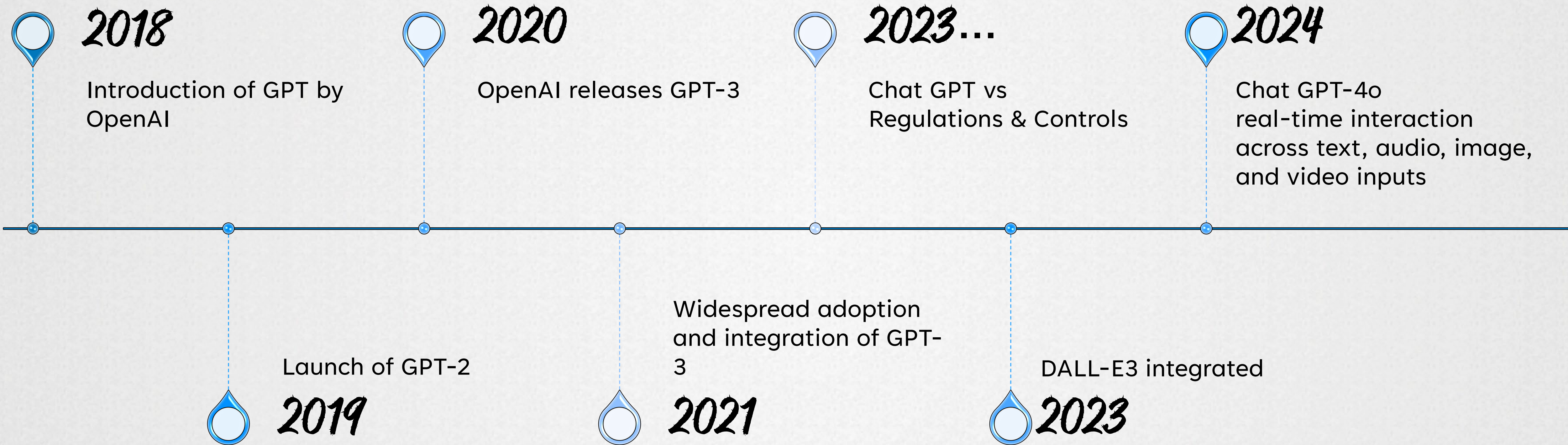
What is AI...

What happens with my data?!

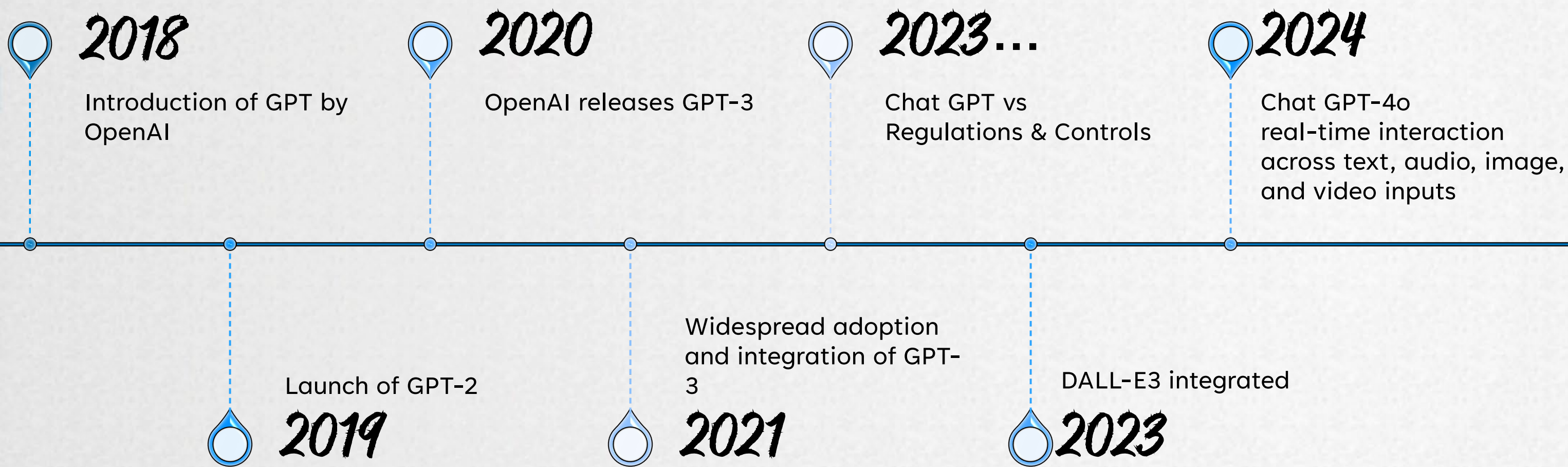
*How does Chat GPT
solve all my testing problems?!*



Evolution of Chat GPT



Evolution of Chat GPT



June 11th: Elon Musk threatens Apple

End of year: \$1.000.000.000



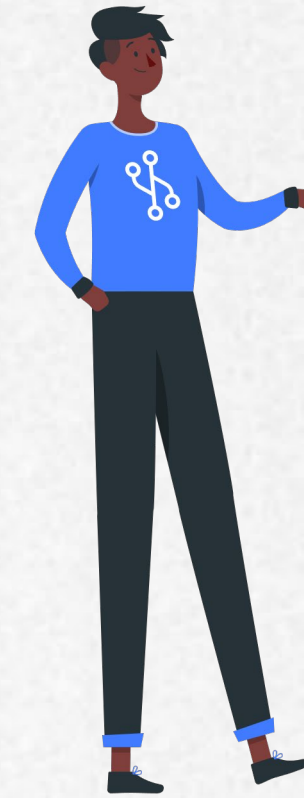


Chat GPT Overview



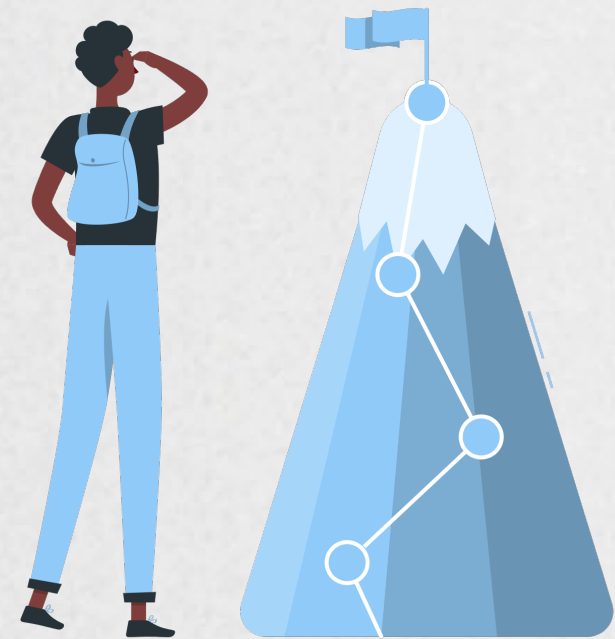
Human-like Text Generation

Ability to produce coherent and contextually relevant text.



Versatility

From generating test cases to answering queries, its applications are vast.



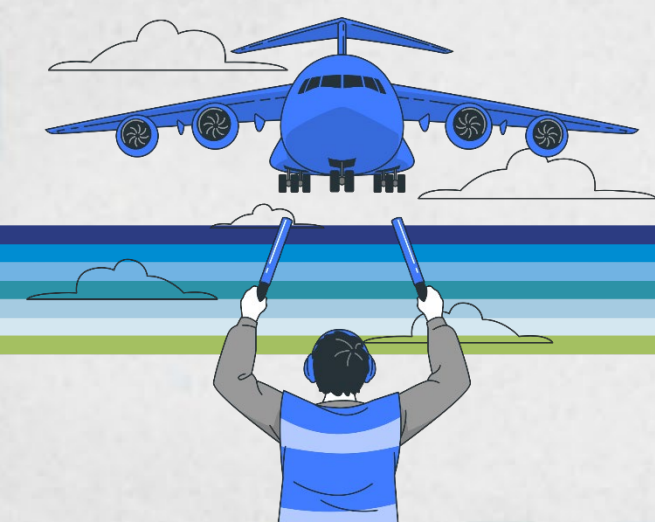
Scalability

Can handle large volumes of requests, making it suitable for extensive testing scenarios.



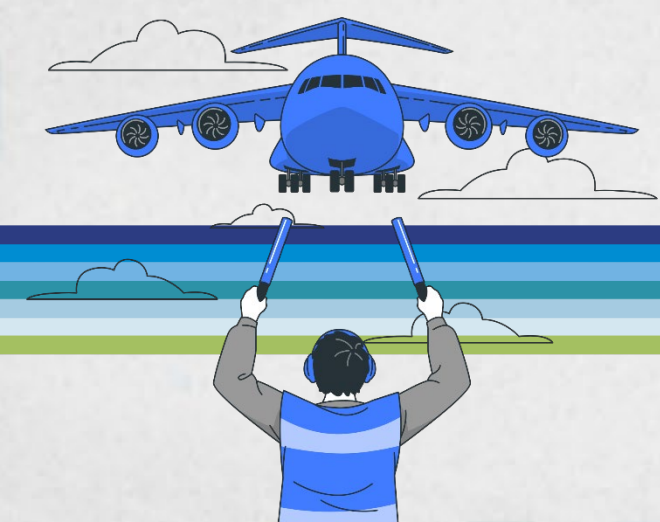
Continuous Learning

Adapts and improves over time, ensuring up-to-date responses.



Bridging Communication Gaps:

Translates technical jargon into accessible language
-> fostering better collaboration.



Why is it so difficult?!



ChatGPT's Role

Automated Generation:

ChatGPT can quickly generate test cases based on provided requirements.

Diverse Scenarios:

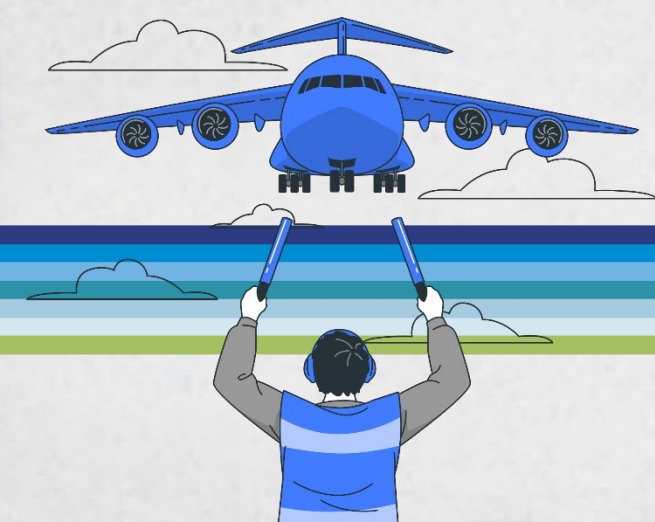
It can cover a wide range of scenarios, ensuring comprehensive testing.

Clear & Understandable:

Test cases produced are free from jargon and easily understandable by all stakeholders.

Adaptive Learning:

ChatGPT refines its outputs based on feedback, ensuring continuous improvement.



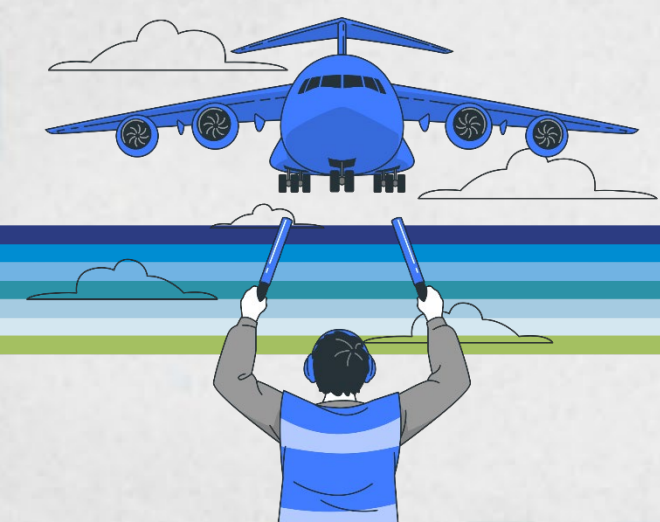


Examples

A test case “classic”

- User login verification

Business Feedback vs Test Case from ChatGPT



A test case “classic”

- User login verification

Business Feedback

What to Do:

1. Go to the place on the website or app where it asks you to log in.
2. Put in your email.
3. Type in your password.
4. There should be a button that says 'Log In'. Click on it.

Result:

After you click the button, you should see your main page or dashboard or in your account.

Test Case from ChatGPT

Title: User Login Verification

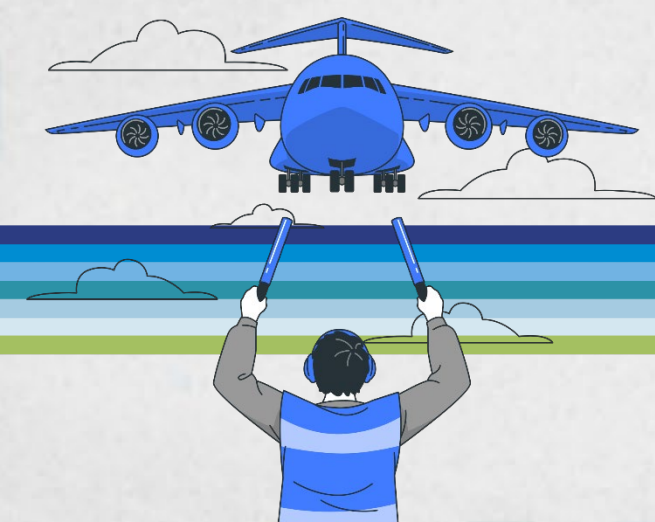
Preconditions: User is registered and has valid credentials.

Steps:

1. Navigate to the application's login page.
2. Enter the user's registered email in the 'Email' field.
3. Enter the user's password in the 'Password' field.
4. Click the 'Login' button.

Expected Result: User is successfully logged in and redirected to the dashboard.

Comments: Ensure the application's server is running and the database connection is active.





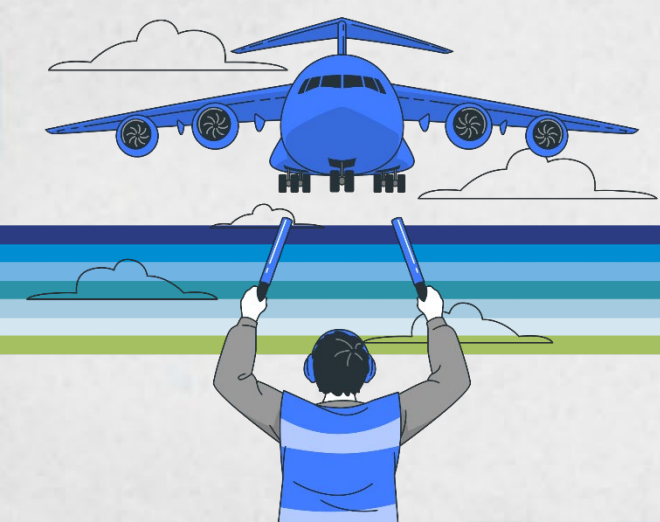
Examples

Bridging the gap

- **Technical Jargon:**
Often creates barriers between technical and non-technical stakeholders.
- **Misunderstandings:**
Ambiguities can lead to misinterpreted requirements or overlooked functionalities.
- **Lack of Engagement:**
Non-technical stakeholders might feel left out of the process due to the technical barrier.



- **Clear Language:**
Translates technical terms into easily understandable language.
- **Interactive Queries:**
Stakeholders can ask questions and get clear answers, reducing misunderstandings.
- **Inclusive Collaboration:**
By simplifying language, all stakeholders, regardless of their technical background, can actively participate in the process.



The 6 prompt components

[task]

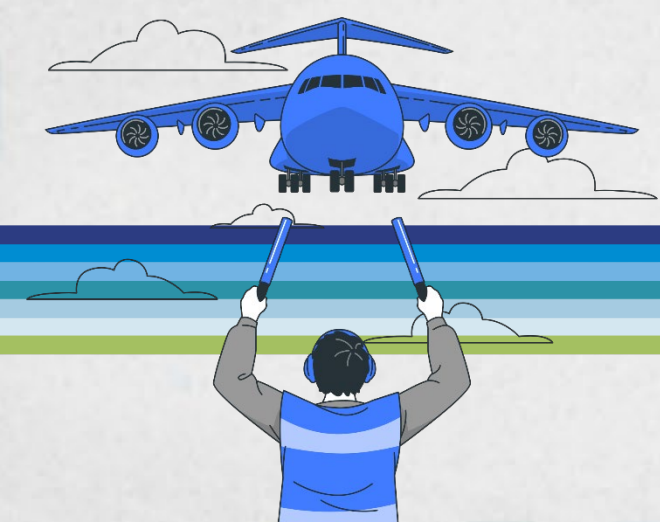
[context]

[example]

[persona]

[format]

[tone]



The 6 prompt components

[task]

Mandatory

[context]

Important

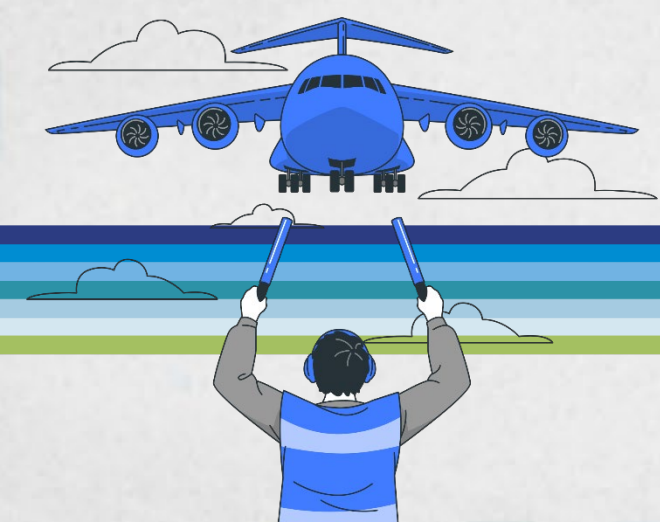
[example]

[persona]

[format]

Nice to have

[tone]



Hi AI,
I need you to create test cases.



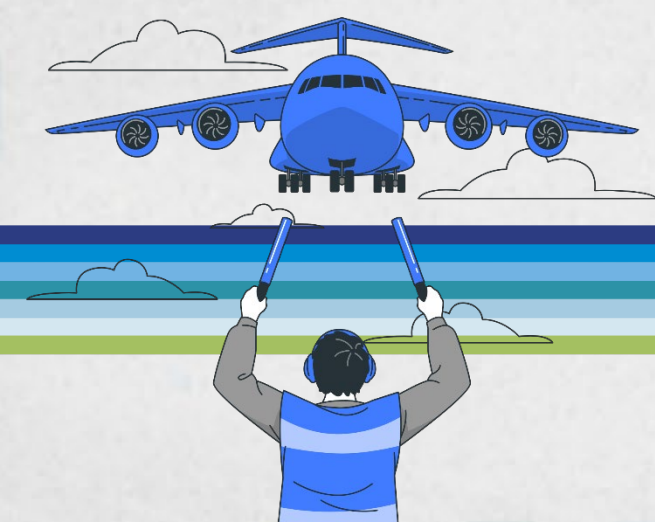
Of course! To create effective test cases, I'll need some details:

1. What application or functionality are we testing?
2. What specific features or components need test cases?
3. Is there any particular type of testing required (e.g., unit testing, integration testing, UI testing)?
4. Any specific requirements or scenarios to consider?

If you provide some information on the application or feature, I can create relevant test cases for you.



[task] ✓



I need you to create test cases for the following requirement:

Product Backlog Item: Password Reset Functionality


As a registered user,
I want to reset my password,
so that I can regain access to my account if I forget my password or
feel that my account security has been compromised.

Acceptance Criteria:

1. When I click on the "Forgot Password" link, I should be directed to the password reset page.
2. I should be able to enter my registered email address to request a password reset.
3. After entering my email and submitting the request, I should receive a confirmation message informing me to check my email.
4. I should receive a password reset email with a unique link that expires in 24 hours.
5. Clicking the link in the email should direct me to a secure page where I can set a new password
6. After setting a new password, I should be redirected to the login page with a message indicating the password change was successful.

[task] ✓

[context] ✓



Certainly! Below are the test cases for the password reset functionality based on the provided requirements:

Test Case 1: Navigation to Password Reset Page

Description: Verify that the user is directed to the password reset page when clicking on the "Forgot Password" link.

- Preconditions:** User is on the login page.
- Steps:**
 - Click on the "Forgot Password" link.
- Expected Result:** The user is navigated to the password reset page.

Test Case 2: Enter Registered Email Address for Password Reset

Description: Verify that the user can enter their registered email address to request a password reset.

- Preconditions:** User is on the password reset page.

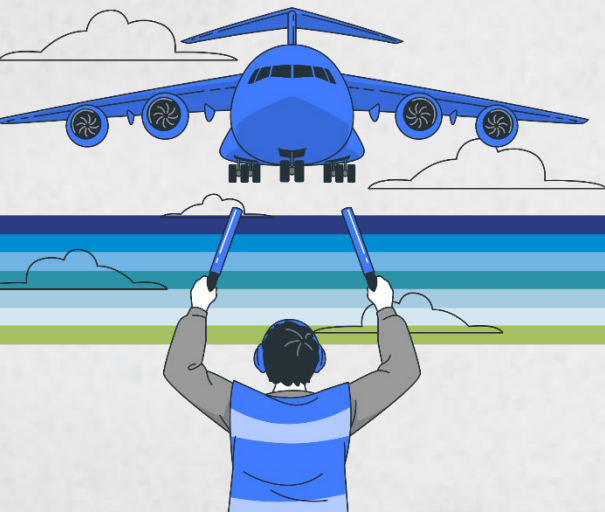
[task]

[context]

[example]

[format]

Could you please provide the test cases in table format?
The column titles are: Test Case ID, Title, Preconciliation, Steps, Expected Result





Sure! Here are the test cases in a table format:

Test Case ID	Title	Precondition	Steps	Expected Result
TC01	Navigation to Password Reset Page	User is on the login page	1. Click on the "Forgot Password" link.	The user is navigated to the password reset page.
TC02	Enter Registered Email Address for Password Reset	User is on the password reset page	1. Enter a registered email address in the email input field. 2. Submit the request.	The system accepts the email address and processes the request.
TC03	Confirmation Message after Submitting Password Reset	User is on the password reset page with an email	1. Submit the password reset request.	A confirmation message is displayed to check email for further instructions.
TC04	Receive Password Reset Email with Unique Link	User has submitted a password reset request	1. Check the registered email inbox. 2. Open the password reset email.	The email contains a unique password reset link that expires in 24 hours.
TC05	Link Directs to Secure Page for New Password	User has received the password reset email	1. Click the password reset link in the email.	The user is directed to a secure page to set a new password.
TC06	Set New Password and Redirect to Login Page	User is on the secure page to set a new password	1. Enter and confirm a new password. 2. Submit the new password.	The user is redirected to the login page with a message indicating

[task]

[context]

[example]

[format]



Good job thank you.
I have another requirement for you. Unfortunately, it is very technical. I need you to create test cases that are easy to understand for business users that have no technical experience.

Here is the new requirement:

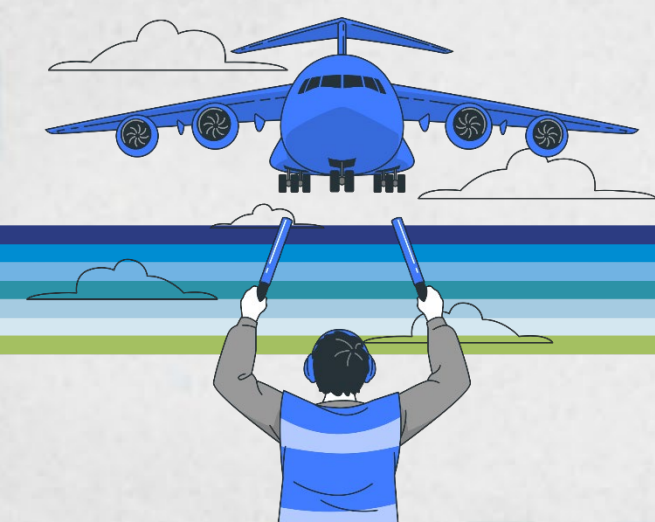
[task] ✓

[context] ✓

[example] ✓

[format] ✓

[tone] ✓





Functional Requirement: Database Replication in a Distributed System

Title: Implementing Asynchronous Replication for Horizontal Database Scalability

Objective:

To ensure data consistency, availability, and fault tolerance in our distributed system, we need to implement asynchronous replication across multiple database nodes. This will allow us to achieve horizontal scalability and ensure that data is always available, even if one or more nodes fail.

Technical Details:

1. Replication Type: Asynchronous Master-Slave replication.
2. Database System: PostgreSQL 13.2.
3. Network Topology: All database nodes are located within the same Virtual Private Cloud (VPC) but spread across three different availability zones.
4. Data Consistency: Implement the use of Write-Ahead Logging (WAL) to ensure that all changes to the master database are logged and replicated to slave nodes.
5. Failover Mechanism: In the event of a master node failure, one of the slave nodes should be promoted to master status. This should be facilitated by a combination of health checks and automated scripts.
6. Load Balancing: Read requests should be distributed across slave nodes to distribute the load and optimize performance. Use Pgpool-II for connection pooling and load balancing.
7. Security: All data transferred between nodes should be encrypted using TLS. Additionally, only nodes within the VPC should be able to communicate with each other, ensuring a secure environment.

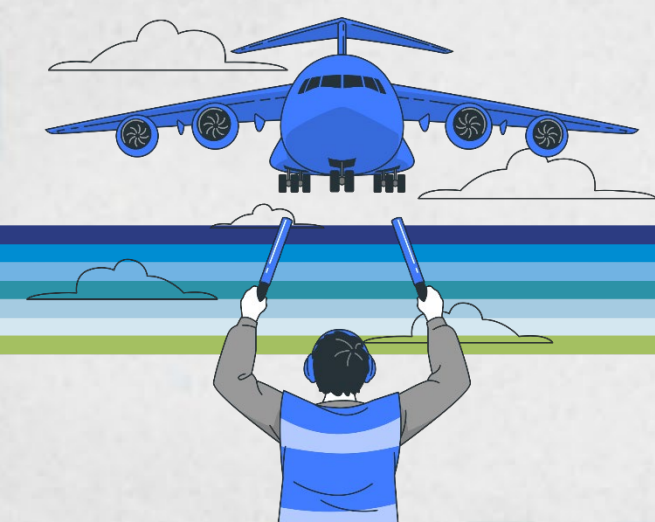
[task]

[context]

[example]

[format]

[tone]



TC ID	Title	Precondition	Steps	Expected Result
TC01	Data Replication	Data is present in the master node.	1.Add a new record to the master database. 2. Wait for a short duration.	The new record should appear in all slave databases, ensuring data is replicated.
TC02	Master Node Failure and Recovery	All nodes are operational.	1.Simulate a failure of the master node. 2. Monitor the system's response.	One of the slave nodes should automatically become the new master, ensuring continuous data availability.
TC03	Load Distribution for Read Operations	Multiple slave nodes are operational and contain data.	1.Execute multiple read requests simultaneously.	Read requests should be distributed across slave nodes, ensuring faster data retrieval.
TC04	Data Security During Transfer	Data is being replicated between nodes.	1. Monitor the data transfer between nodes.	Data transfer should be encrypted, ensuring data security.
TC05	Restricted Node Communication	Nodes are set up in the VPC.	1.Attempt to establish a connection to a node from outside the VPC.	Connection should be denied, ensuring only nodes within the VPC can communicate with each other.
TC06	Data Consistency with Write-Ahead Logging	Data is present in the master node.	1.Make changes to the master database. 2.Check the logs.	All changes should be logged, ensuring data consistency during replication.
TC07	Lag in Data Replication	Data is continuously being added to the master node.	1.Add a new record to the master database. 2.Measure the time taken for it to appear in slave nodes.	The lag should be minimal, ensuring timely data replication.
TC08	Load Balancing with Pgpool-II	Pgpool-II is set up for connection pooling and load balancing.	1. Monitor the distribution of read requests.	Requests should be evenly distributed across slave nodes, optimizing performance.

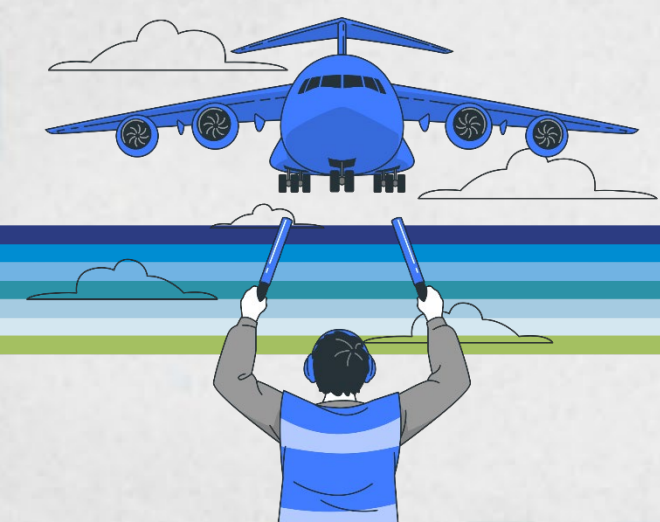


Last but not least...

#1 Create a story

- Give as much detail as you want the answer to be
- Ask yourself “what piece of information would I need to...”
- Define Governance and Rules

#2 Play around

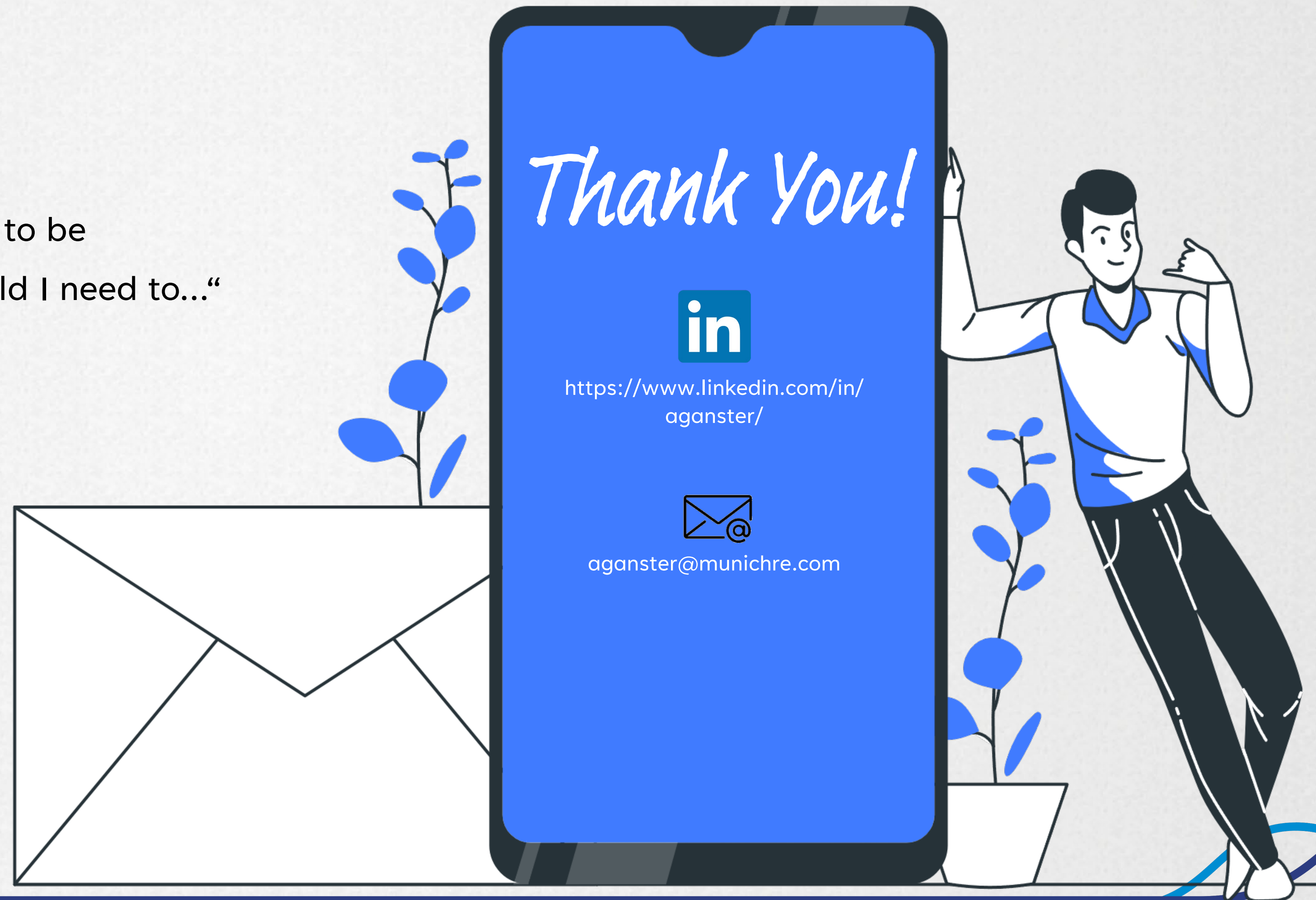


Last but not least...

#1 Create a story

- Give as much detail as you want the answer to be
- Ask yourself “what piece of information would I need to...”
- Define Governance and Rules

#2 Play around



Thanks to our sponsors

PLATINUM SPONSOR



WE LIVE IT



GOLD SPONSOR

