LECTURE 3

Software requirements Engineering

JIRA, A Generic Process for Requirements Engineering

Instructor:
Dr. Natalia Chaudhry,
Assistant Professor, PUCIT, University of the Punjab, Lahore.

JIRA

- Jira is a powerful tool for requirement engineering, and it offers various features to manage project requirements efficiently.
- provides free license for academic projects.
- used in Bugs, Issues and Change Request Tracking.
- can be used in Help desk, Support and Customer Services to create tickets and track the resolution and status of the created tickets.
- useful in Project Management, Task Tracking and Requirement Management.
- is very useful in Workflow and Process management.

JIRA - Core Features

Boards

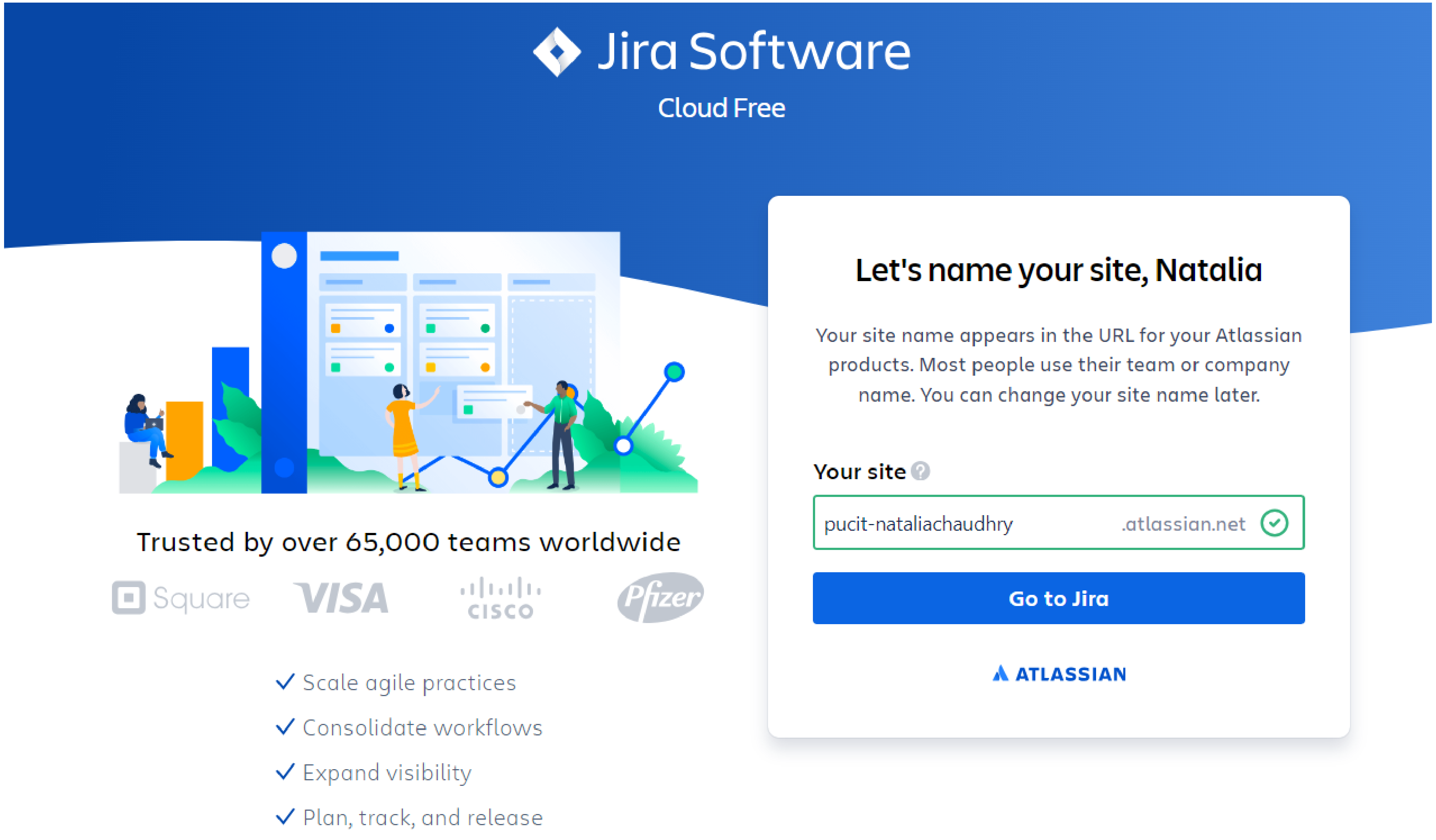
- JIRA supports Scrum and Kanban boards.
- These boards provide an immediate snapshot of the project to the team.
- Helps to quickly review the progress of the project and see the status of the individual tasks.
- Kanban and Scrum are project management methodologies that complete project tasks in small increments and emphasize continuous improvement.
- While Kanban is centered around visualizing tasks and continuous flow, Scrum is more about implementing timelines for each delivery cycle and assigning set roles.

- The heart of the Kanban method is the Kanban board—physical or digital—in which phases of the project are divided into columns. Tasks are written on cards that progress from one column to the next, until the task is completed.
- Scrum is an Agile methodology designed for complex projects where it is frequently necessary to adapt to change.
- Scrum is based on short development cycles called sprints, which generally last from one to four weeks.

Step 1: Setting up Jira Project

1. **Create a Jira Account**: If you haven't already, sign up for a Jira account. You can use the cloud version or set it up on your own server.
2. **Create a New Project**: Once logged in, create a new project. Choose the project type based on your requirements. For requirement engineering, the "Scrum" or "Kanban" template can be suitable.
3. **Define Issue Types**: Customize your project's issue types to include requirements. Typically, you can add a "User Story" or "Requirement" issue type.

https://www.atlassian.com/software/jira/free



Let's name your site, Natalia

Your site name appears in the URL for your Atlassian products. Most people use their team or company name. You can change your site name later.

Your site

pucit-nataliachaudhry .atlassian.net

Trusted by over 65,000 teams worldwide

✓ Scale agile practices
✓ Consolidate workflows
✓ Expand visibility
✓ Plan, track, and release

Where would you like to start?

Select one template to get started, it's easy to change later

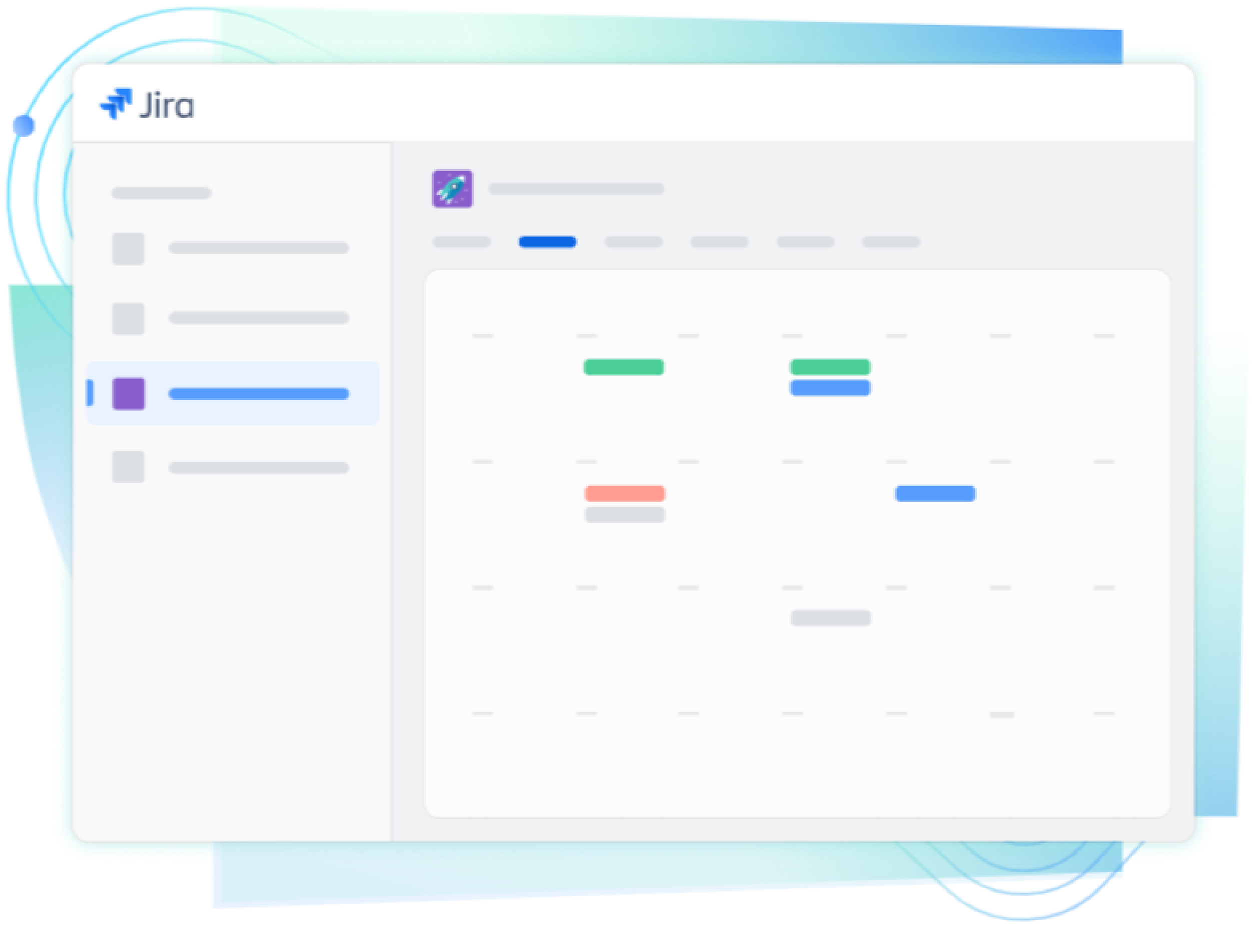
For software development
Plan sprints and build product roadmaps in agile

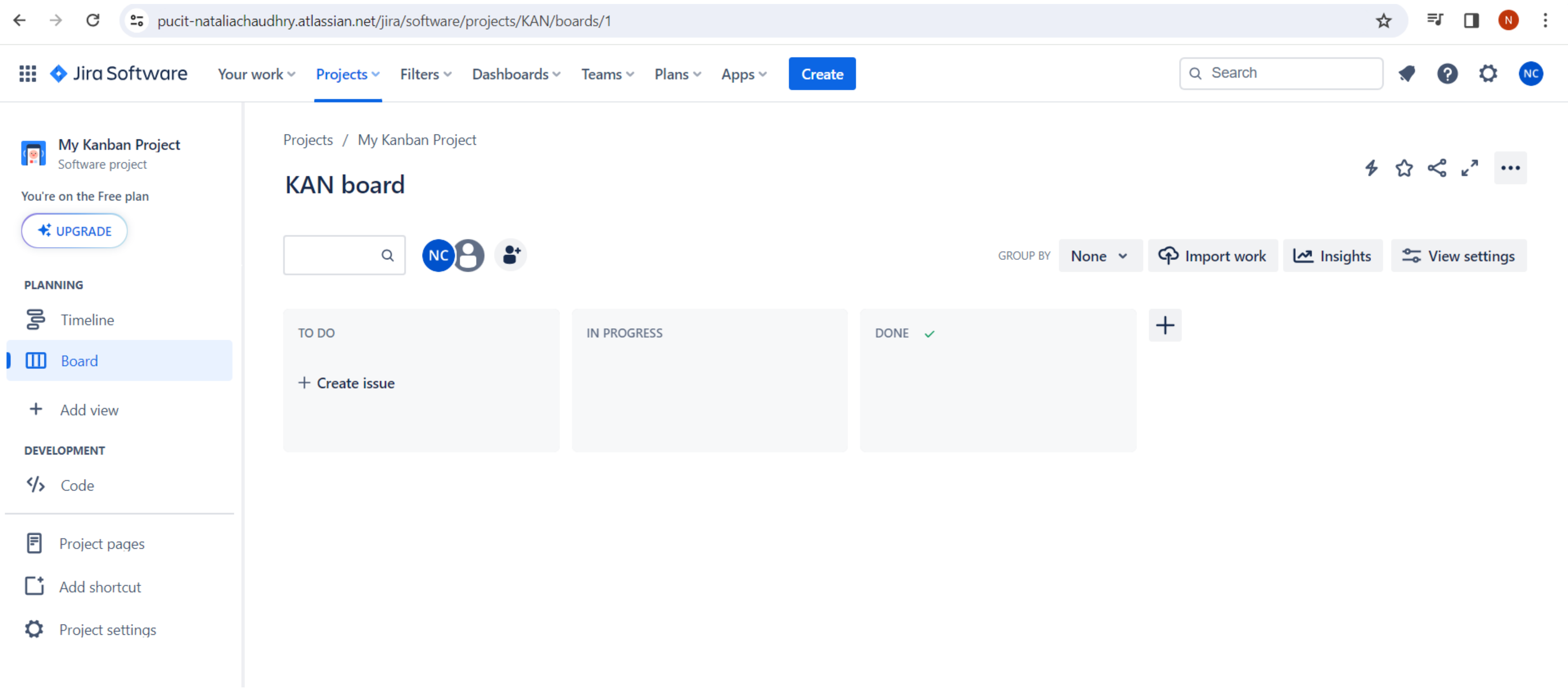
Kanban | Scrum

For business
Manage and deliver work across all types of teams

Project management

Finish





pucit-nataliachaudhry.atlassian.net/jira/software/projects/KAN/boards/1

My Kanban Project
Software project

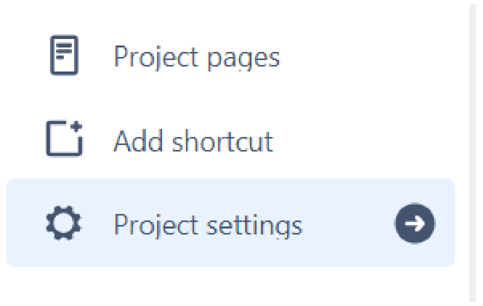
You're on the Free plan

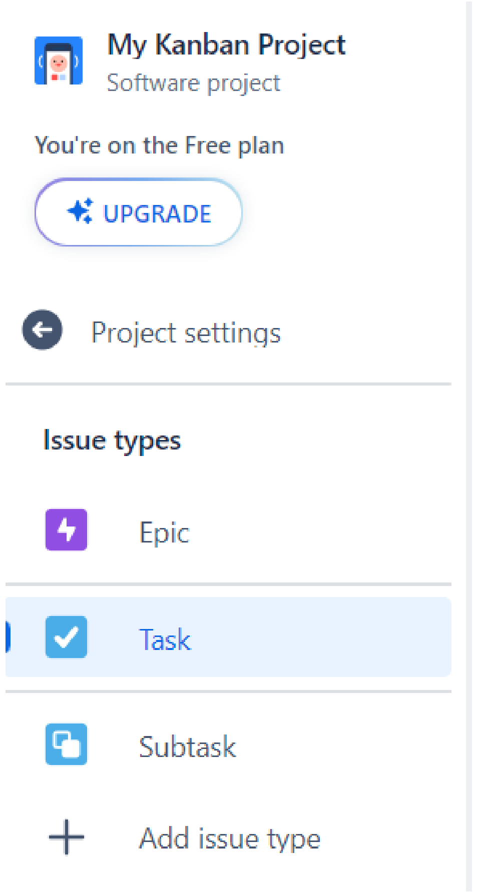
Projects / My Kanban Project

KAN board

To Generate A New User Story, Follow Five Simple Steps:

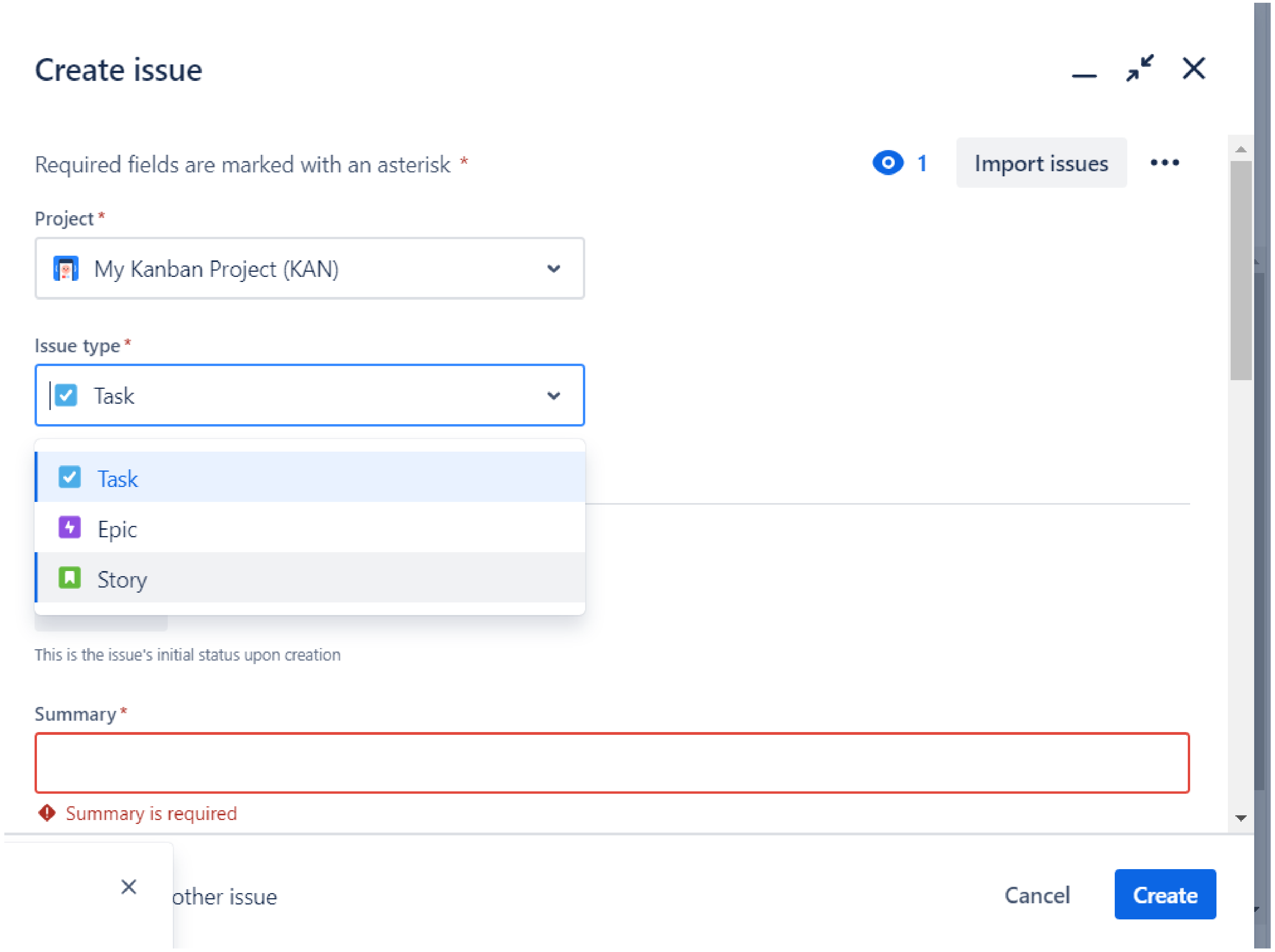
Project settings>Issue types > +add issue type





- Go to the new issue creation screen in your project.*
- Create a new issue and select the desired Project from the menu.*
- Set the Type – 'Story.' *
- Add some words to the summary field.*
- Fill in the description field if necessary.

You can then use the summary field to fill it with the user story itself. You will see it on the new issue creation screen.


Create issue

Required fields are marked with an asterisk *

Project *

My Kanban Project (KAN)

Issue type *

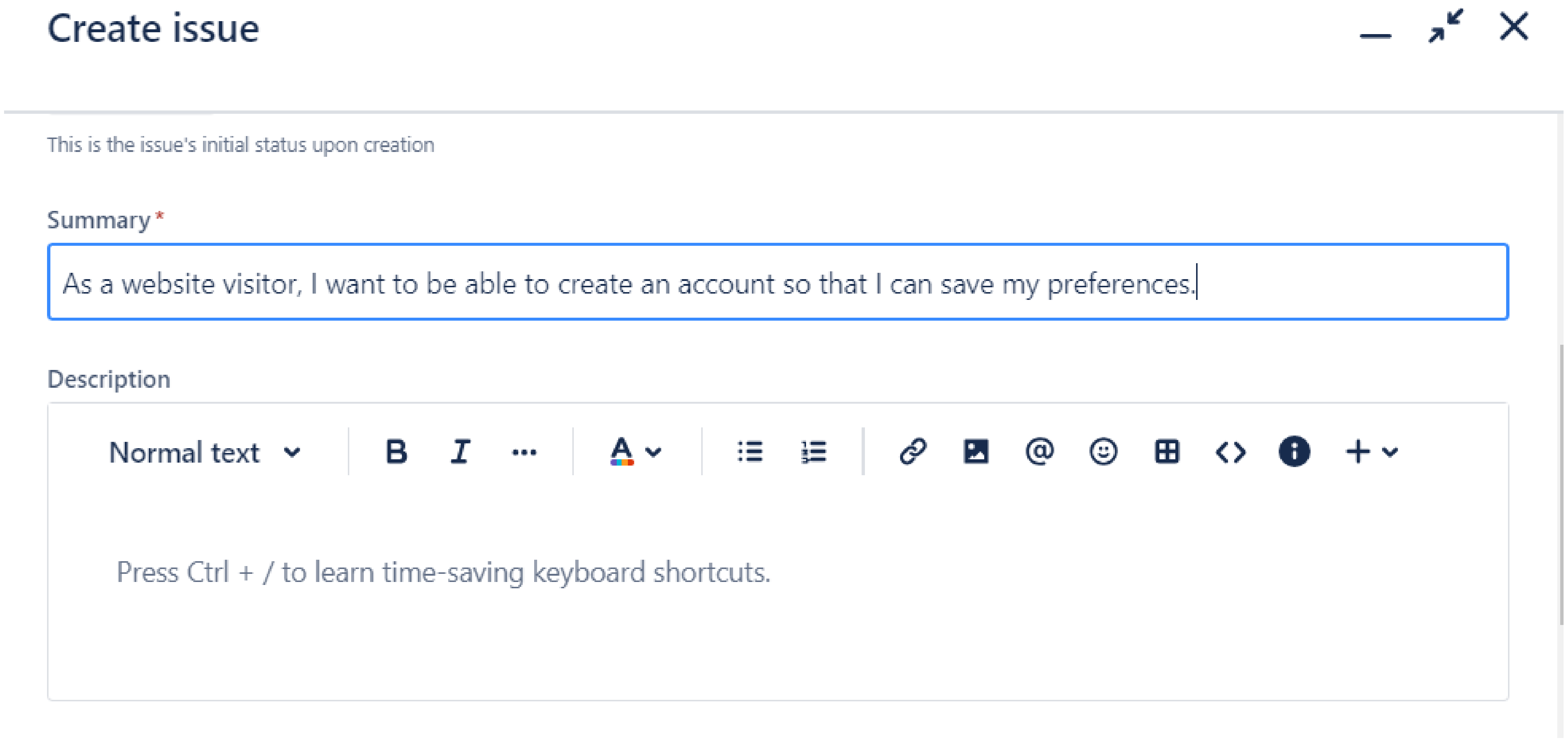
Task

This is the issue's initial status upon creation

Summary *

Summary is required

As a website visitor, I want to be able to create an account so that I can save my preferences.



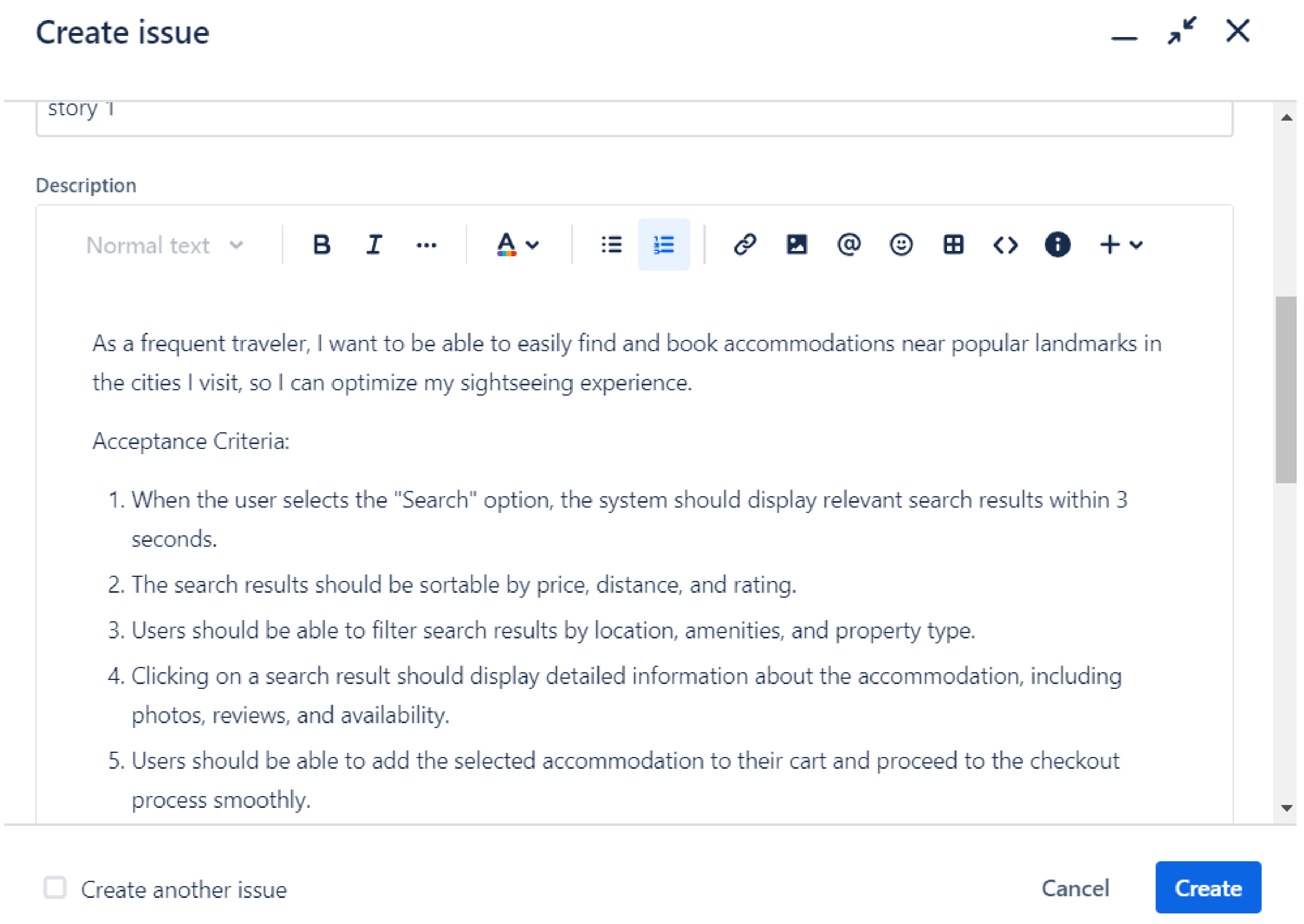
This is the issue's initial status upon creation

Summary*

As a website visitor, I want to be able to create an account so that I can save my preferences.

Description

Press Ctrl + / to learn time-saving keyboard shortcuts.

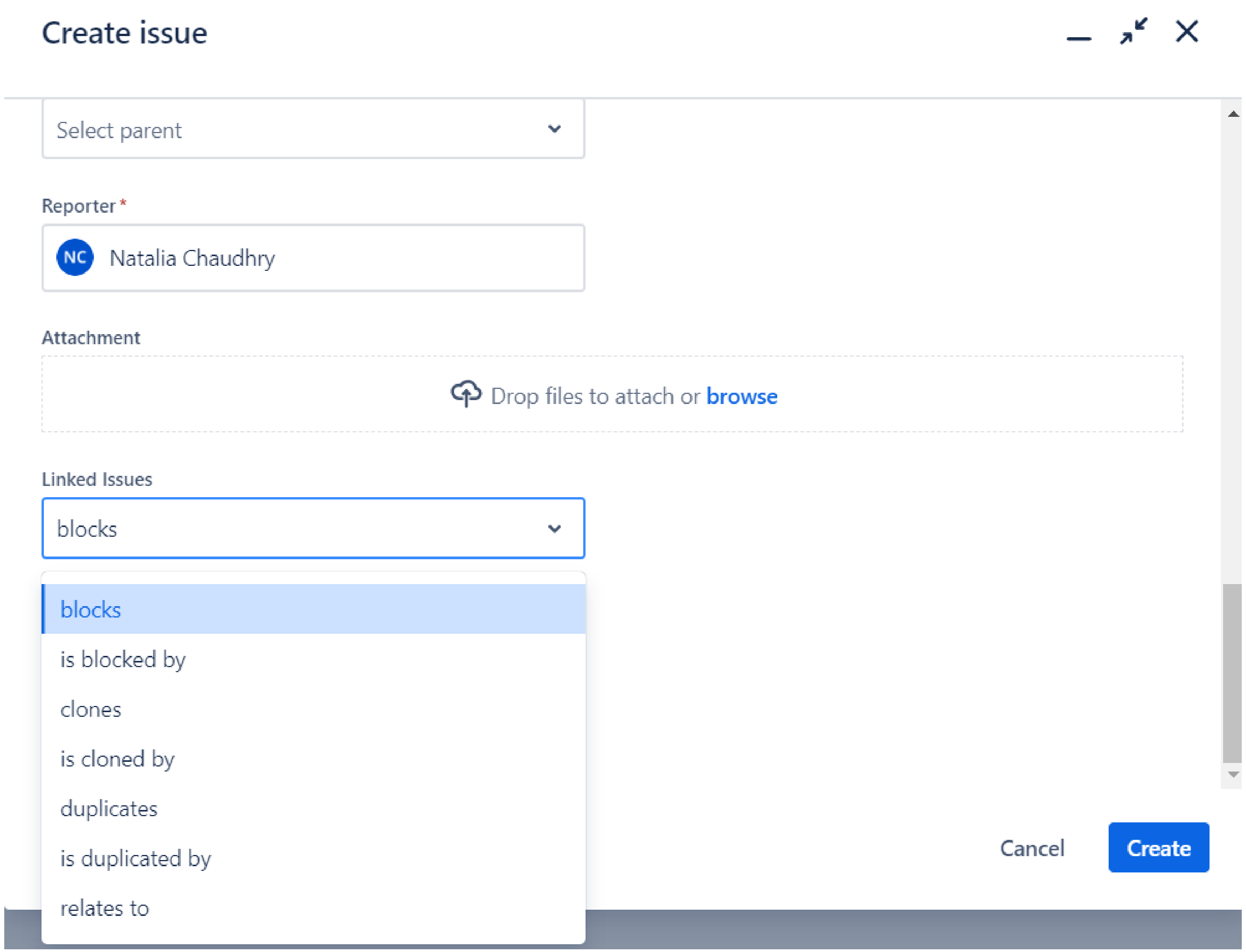


As a frequent traveler, I want to be able to easily find and book accommodations near popular landmarks in the cities I visit, so I can optimize my sightseeing experience.

Acceptance Criteria:

1. When the user selects the "Search" option, the system should display relevant search results within 3 seconds.
2. The search results should be sortable by price, distance, and rating.
3. Users should be able to filter search results by location, amenities, and property type.
4. Clicking on a search result should display detailed information about the accommodation, including photos, reviews, and availability.
5. Users should be able to add the selected accommodation to their cart and proceed to the checkout process smoothly.

In Jira, **"Linked Issues"** refer to the connections between different issues within a project or across multiple projects. These connections can be of various types, and they help to establish relationships and dependencies between issues.



Create issue

Select parent

Reporter *

Natalia Chaudhry

Attachment

Drop files to attach or **browse**

Linked Issues

blocks

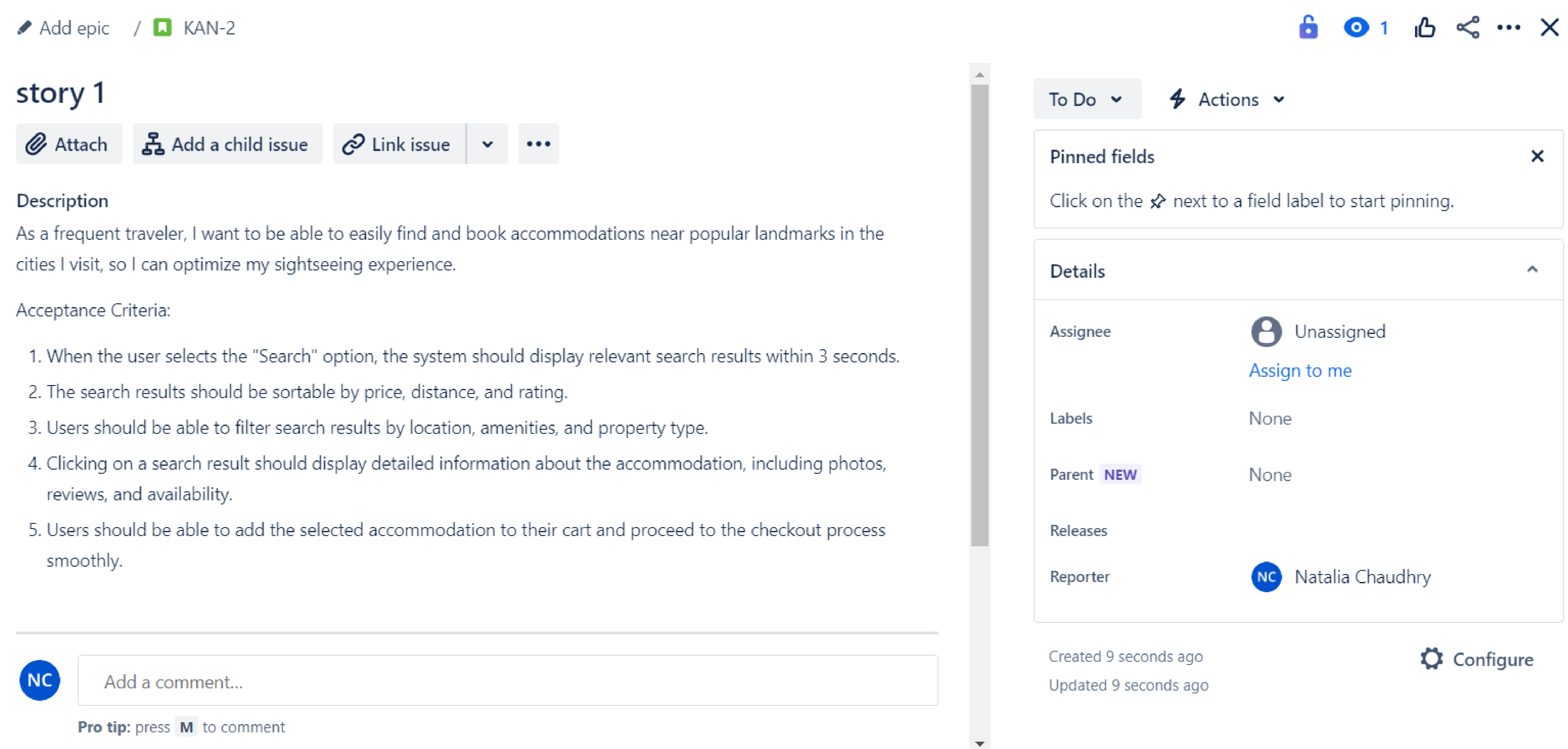
- blocks
- is blocked by
- clones
- is cloned by
- duplicates
- is duplicated by
- relates to

Cancel

Create

There are several types of links that can be established between issues:

1. **Blocks/Blocked By**: Indicates that one issue cannot be completed until another issue is resolved. For example, if Issue A is blocking Issue B, then Issue B cannot be completed until Issue A is resolved.
2. **Relates to**: Indicates a general relationship between two issues, such as them being related in some way, but not necessarily dependent on each other for completion.
3. **Duplicates/Duplicated By**: Indicates that one issue is a duplicate of another issue, or that one issue has been created as a result of another issue being duplicated.
4. **Is Parent Of/Is Child Of**: Indicates a hierarchical relationship between issues, where one issue is a parent of another issue, or one issue is a child of another issue.
5. **Causes/Is Caused By**: Indicates that one issue is the cause of another issue, or that one issue is a result of another issue.



Add epic / KAN-2

story 1

Description

As a frequent traveler, I want to be able to easily find and book accommodations near popular landmarks in the cities I visit, so I can optimize my sightseeing experience.

Acceptance Criteria:

1. When the user selects the "Search" option, the system should display relevant search results within 3 seconds.
2. The search results should be sortable by price, distance, and rating.
3. Users should be able to filter search results by location, amenities, and property type.
4. Clicking on a search result should display detailed information about the accommodation, including photos, reviews, and availability.
5. Users should be able to add the selected accommodation to their cart and proceed to the checkout process smoothly.

Add a comment...

Pro tip: press M to comment

Pinned fields

Click on the next to a field label to start pinning.

Details

Assignee	Unassigned Assign to me
Labels	None
Parent NEW	None
Releases	
Reporter	Natalia Chaudhry

Created 9 seconds ago
Updated 9 seconds ago

Linking stories

let's consider a scenario where one user story depends on the completion of another user story.
Scenario: In a project to develop an e-commerce website, the team is tasked with implementing a shopping cart feature. However, before users can add items to the shopping cart, they need to be able to browse products and select items they want to purchase.

User Story 1: As a shopper, I want to be able to browse products so that I can see what items are available for purchase.
User Story 2: As a shopper, I want to be able to add items to my shopping cart so that I can purchase them.

Dependency: User Story 2 (adding items to the shopping cart) depends on the completion of User Story 1 (browsing products), because users need to browse and select items before they can add them to the shopping cart.

1. Navigate to "User Story 2" (adding items to the shopping cart) in Jira.
2. Click on the "Link" option.
3. Select "Dependency" as the link type.
4. Search for "User Story 1" (browsing products) and select it.
5. Optionally, add a comment explaining the dependency.
6. Save the link.

Now, when viewing User Story 2, it will show that it has a dependency on User Story 1. This helps the team understand the order in which tasks need to be completed and ensures that they prioritize their work effectively.

Notifications

- An email can be sent for a particular task to the users.
- Voting and watching features to keep an eye on the progress for the stakeholders.
- Use **@mention** to get the attention of a specific team member at Comments/Description.
- User will instantly notify if something is assigned or if any feedback is required.

Attachments

As a software developer working on a new feature for a messaging application, I need to review the design mockups provided by the UX/UI team to ensure that the implementation aligns with the intended user experience.

User Story:

- **Title**: Review Design Mockups for Messaging Feature
- **As a**: Software Developer
- **I want to**: Review the design mockups provided by the UX/UI team
- **So that**: I can ensure that the implementation aligns with the intended user experience

feature.

Acceptance Criteria:

1.The UX/UI team attaches design mockups for the messaging feature to this user story in Jira.

2.I review the attached design mockups to understand the visual and functional requirements of the feature.

3.If there are any discrepancies between the design mockups and the implementation, I raise them with the UX/UI team for clarification and resolution.

4.Once the design review is complete and any necessary adjustments are made, I proceed with the implementation of the messaging

Attachment Example:

•The UX/UI team attaches a ZIP file named **"Messaging_Feature_Design_Mockups.zip"** to this user story, containing detailed wireframes and UI designs for the messaging feature.

In this scenario, attachments in Jira play a crucial role in facilitating collaboration between different teams (in this case, the software development team and the UX/UI team). By attaching design mockups to the user story, the UX/UI team provides necessary visual context for the development work. The software developer can then review these attachments directly within the Jira issue, ensuring clear communication and alignment between design and implementation efforts.

Requirement traceability

Requirements traceability is the ability to trace a requirement forwards and backwards in the development lifecycle.

Requirements are traced forward through other development artifacts, including test cases, test runs, and issues. Requirements are traced backward to the source of the requirement, such as a stakeholder or a regulatory compliance mandate.

The purpose of requirements traceability is to verify that requirements are met. It also accelerates development.

Creating Requirements:

•Start by creating a project in Jira and defining your requirements as issues or stories. For example:

- **Requirement**: As a user, I want to be able to sign in to the application using my email and password.
- **Jira Issue**: User Story - US-001

Linking Requirements to Development Tasks:

•Associate each requirement with the development tasks that implement it. This ensures that every requirement is addressed by the development team. For instance:

- **Development Task**: Implement user authentication logic
- **Linking**: Link US-001 to the development task DEV-001

Linking Requirements to Test Cases:

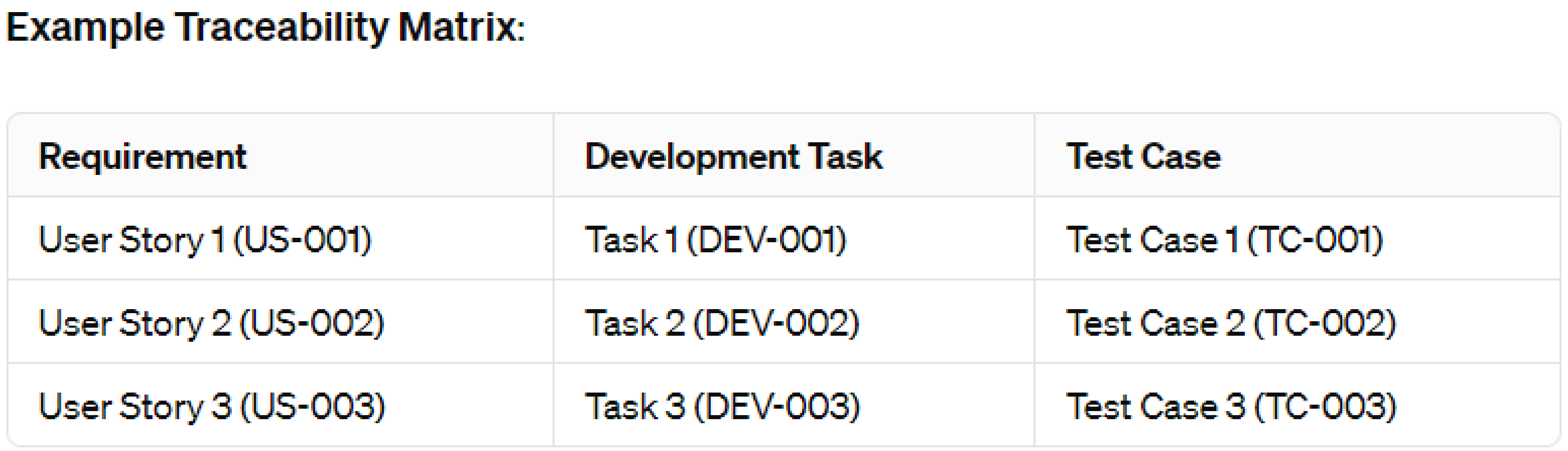
1. Create test cases in Jira and link them to the corresponding requirements to ensure that each requirement is adequately tested. For example:
   1. **Test Case**: Verify user can sign in with correct email and password
   2. **Linking**: Link test case TC-001 to requirement US-001

Tracking Implementation Progress:

1. Use Jira's workflows and status tracking to monitor the progress of requirement implementation and testing. For instance:
   1. Development task DEV-001 transitions from "To Do" to "In Progress" to "Done" as development work completes.
   2. Test case TC-001 transitions from "To Do" to "In Progress" to "Pass" or "Fail" based on the test results.

Maintaining Traceability:

- Regularly review and update the links between requirements, development tasks, and test cases to ensure traceability remains intact throughout the project lifecycle.
- For example, if a new requirement is added or existing ones are modified, update the associated development tasks and test cases accordingly.



Example Traceability Matrix:

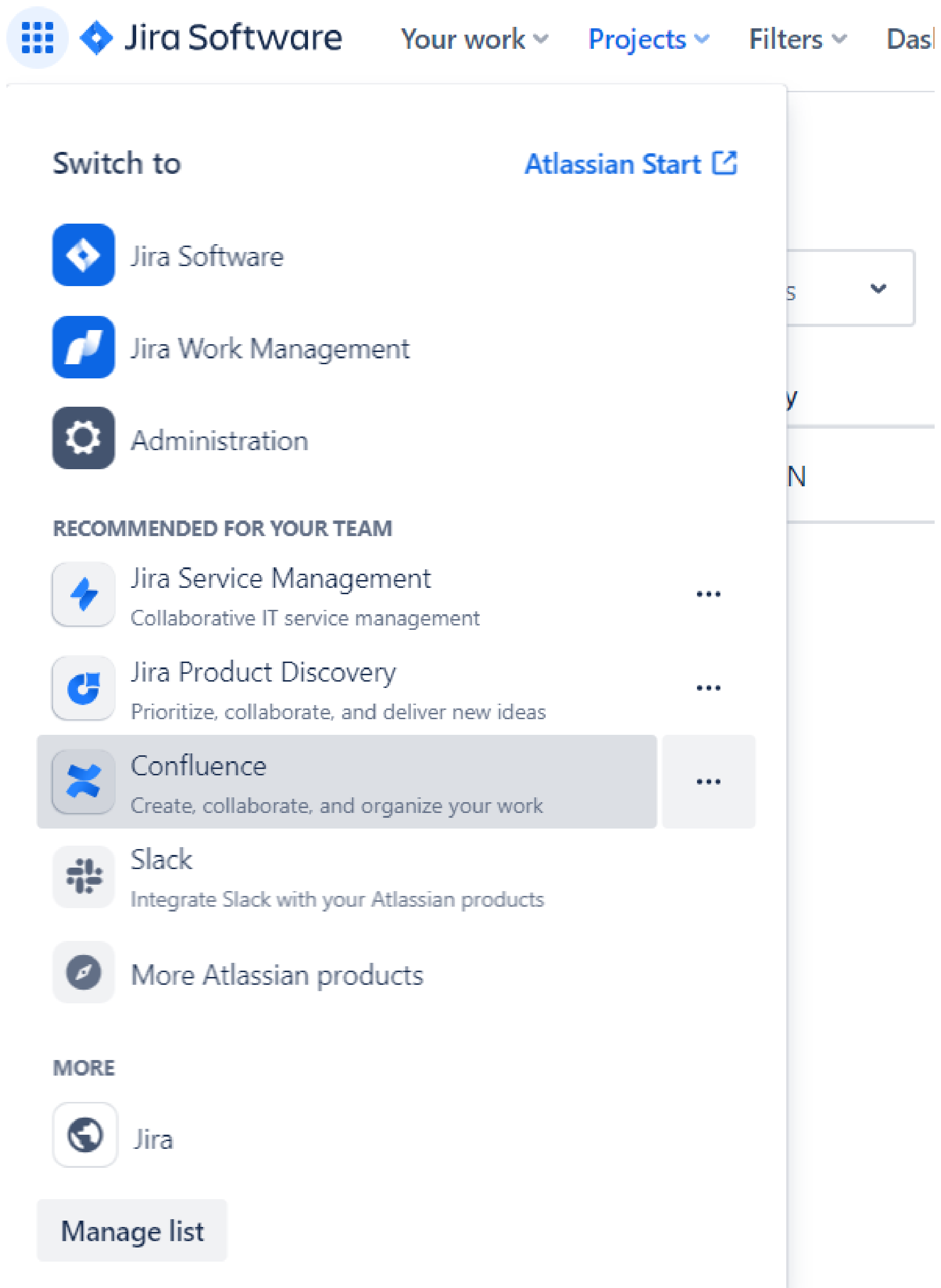
Requirement	Development Task	Test Case
User Story 1 (US-001)	Task 1 (DEV-001)	Test Case 1 (TC-001)
User Story 2 (US-002)	Task 2 (DEV-002)	Test Case 2 (TC-002)
User Story 3 (US-003)	Task 3 (DEV-003)	Test Case 3 (TC-003)

Jira itself doesn't inherently generate traceability matrices in the same way that specialized requirements management tools might. However, you can leverage Jira's features along with plugins or add-ons to create custom solutions for generating traceability matrices.

- **Manual Approach**: You can manually generate a traceability matrix using Jira's advanced search capabilities. For example, you can create JQL (Jira Query Language) queries to retrieve all requirements along with their linked development tasks and test cases.
- **Automation with Plugins**: Explore plugins or add-ons available in the Atlassian Marketplace that offer traceability matrix functionalities. These plugins can automate the generation of traceability matrices based on the relationships between different types of issues in Jira.

Confluence

Confluence Integration: Jira integrates seamlessly with Confluence, Atlassian's documentation tool. You can create detailed requirement documents in Confluence and link them to Jira issues. This allows for more extensive documentation and collaboration on requirements.



Switch to

Atlassian Start

Jira Software

Jira Work Management

Administration

RECOMMENDED FOR YOUR TEAM

Jira Service Management
Collaborative IT service management

Jira Product Discovery
Prioritize, collaborate, and deliver new ideas

Confluence
Create, collaborate, and organize your work

Slack
Integrate Slack with your Atlassian products

More Atlassian products

MORE

Jira

Manage list

Assignment 1

- Explain how **Jira and Confluence** can be utilized to establish and maintain effective end-to-end requirements traceability.
- Must demonstrate how to create requirements
- Cover following aspects: **Create and Edit Pages, Document Collaboration, Project Documentation**, **Task Management**, **Search and Discovery**
  - **Your demonstration should include a video demo with voice over along with manual of steps involved (PDF file)**

Marking will be relative. Perform best to attain best marks!

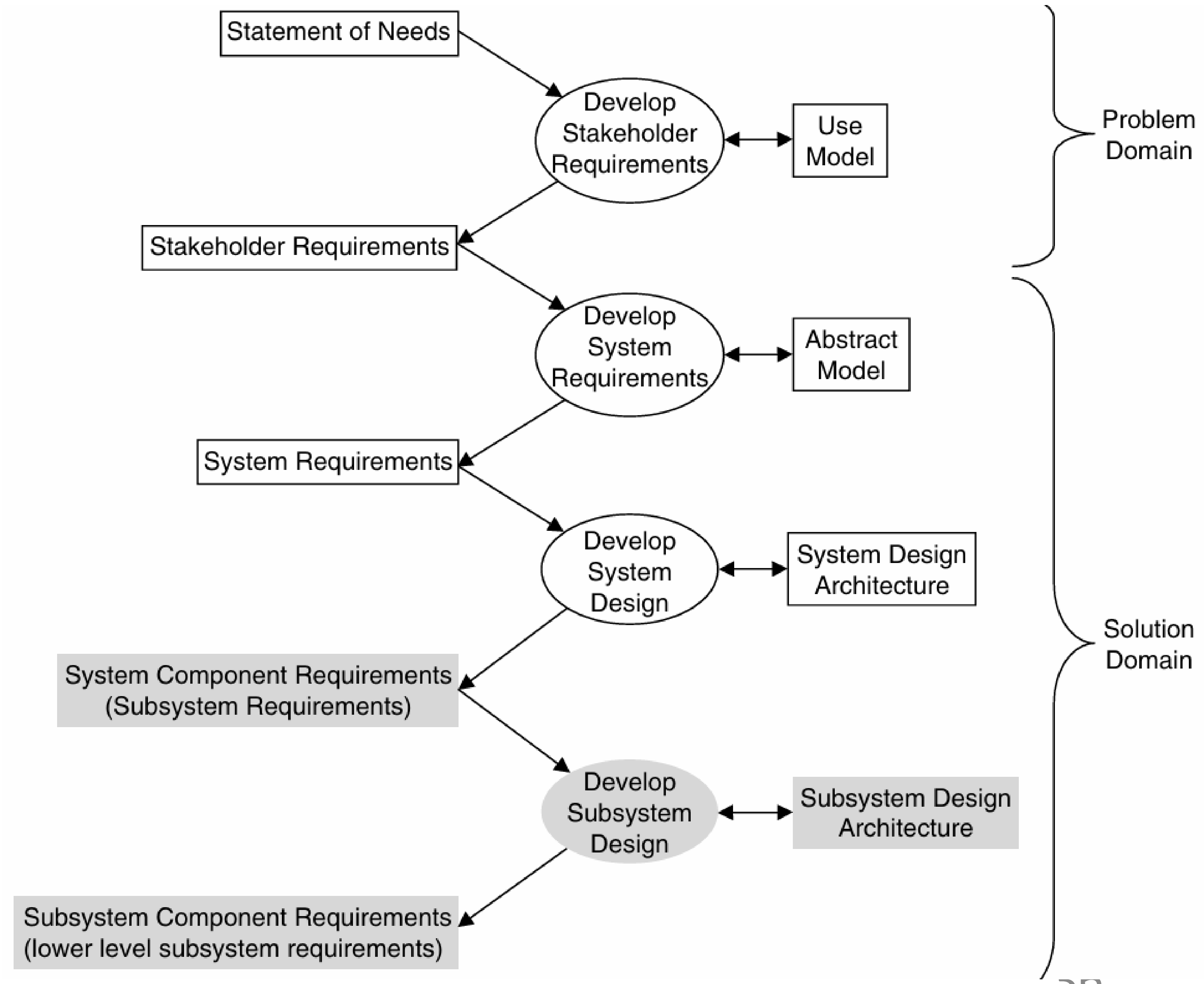
Deadline: 16 feb, 2024, till 11:59 pm

A Generic Process for Requirements Engineering

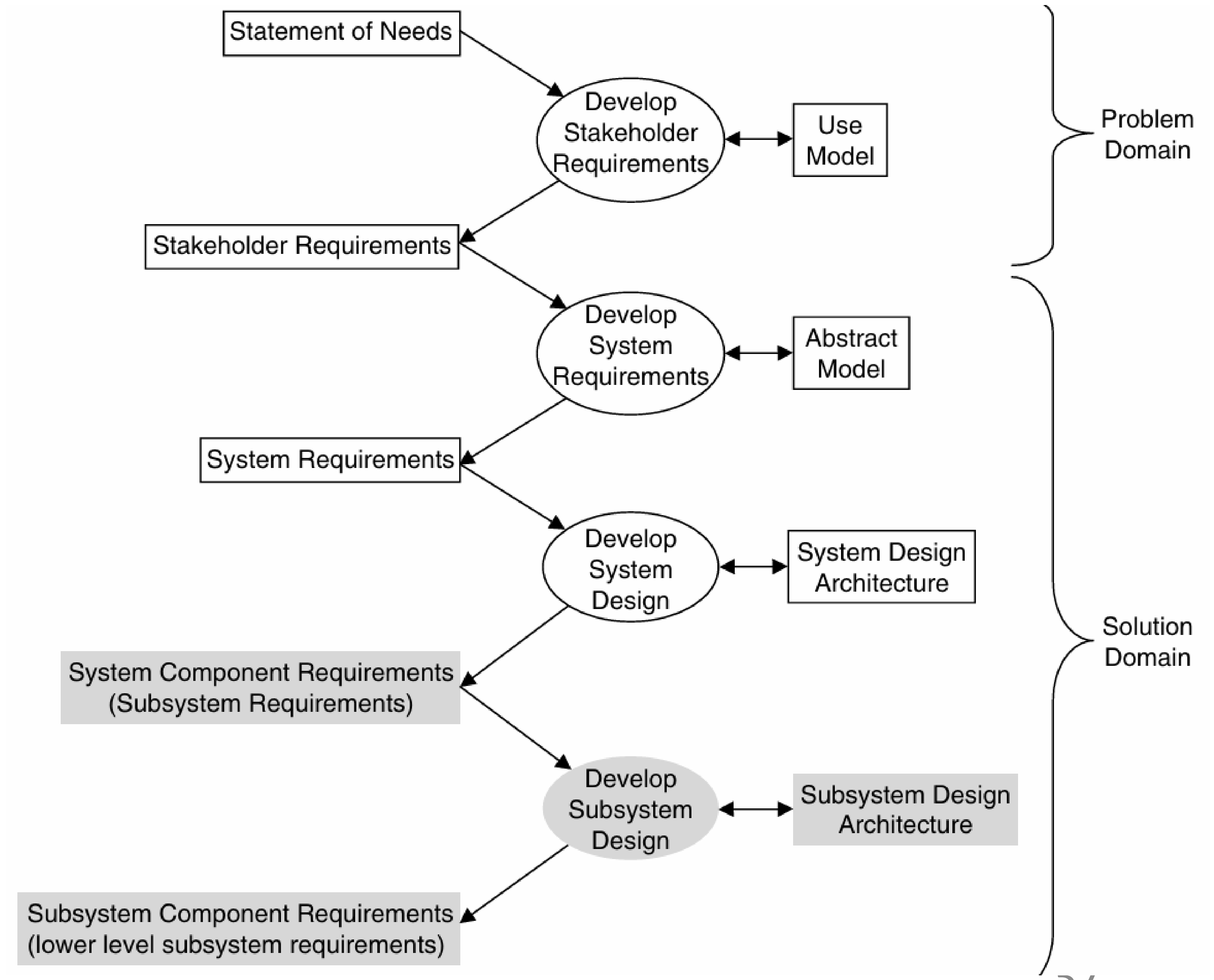
Recommended readings

- Software Requirements, Wiegers K. &Beatty J., 3rd Ed. Microsoft Press, 2013
- **Chapter 2 ,** Requirements Engineering, Elizabeth Hull, Ken Jackson and Jeremy Dick. 3rd Ed, Springer-Verlag London Limited, 2011.

circles (or ovals) represent **<u>processes</u>** and rectangles represent **<u>data or information that is read or produced</u>**. The arrows indicate whether data is read or written.


Statement of Needs
Develop Stakeholder Requirements
Use Model
Stakeholder Requirements
Develop System Requirements
Abstract Model
System Requirements
Develop System Design
System Design Architecture
System Component Requirements (Subsystem Requirements)
Develop Subsystem Design
Subsystem Design Architecture
Subsystem Component Requirements (lower level subsystem requirements)
Problem Domain
Solution Domain

develop stake holder requirements process takes the **<u>statement of needs</u>** and produces the **<u>stake holder requirements</u>**. It also creates and reads a **<u>use model</u>**.



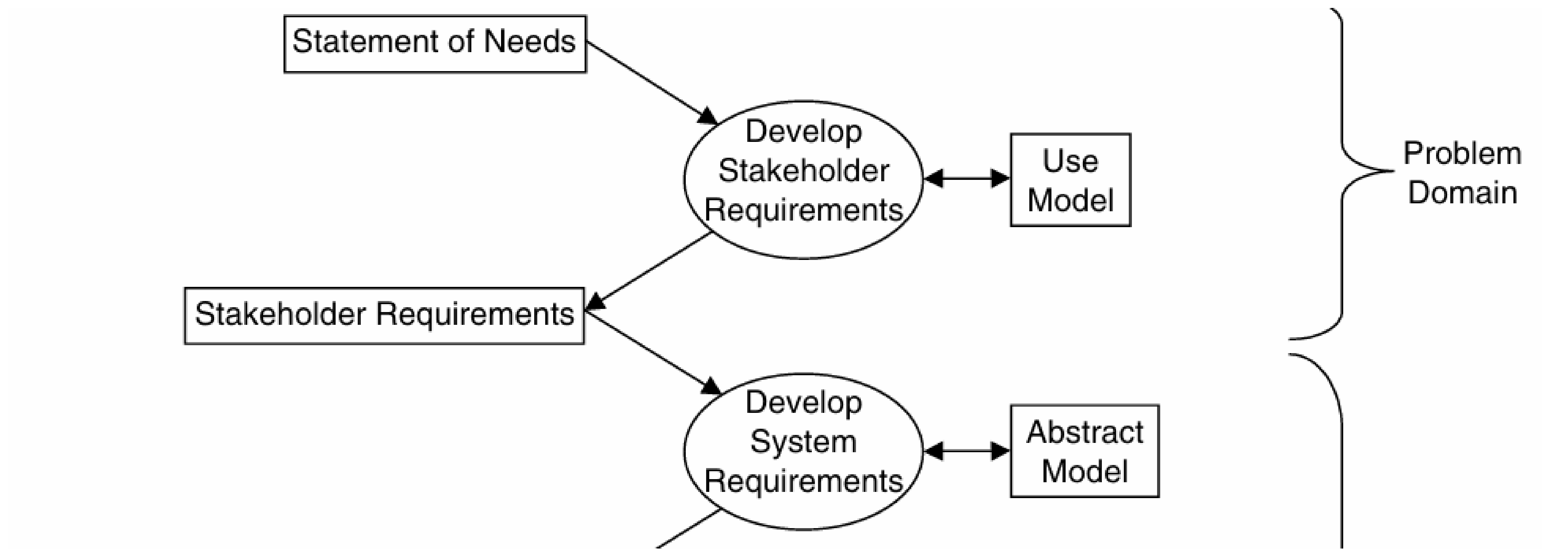
Statement of Needs
Develop Stakeholder Requirements
Use Model
Stakeholder Requirements
Develop System Requirements
Abstract Model
System Requirements
Develop System Design
System Design Architecture
System Component Requirements (Subsystem Requirements)
Develop Subsystem Design
Subsystem Design Architecture
Subsystem Component Requirements (lower level subsystem requirements)
Problem Domain
Solution Domain

A statement of needs, often referred to as a needs statement or needs assessment, is a document that outlines the requirements, desires, or problems that need to be addressed in a particular context or project.
It serves as a roadmap for planning and decision-making.

"Use model" typically refers to a framework or conceptualization that outlines how a product, service, or system is intended to be used by its target audience. It encompasses various aspects such as user behavior, interaction patterns, preferences, and needs. The use model helps designers, developers, and stakeholders understand how users are expected to engage with the product or service.
This may include use case diagrams, sequence diagrams, or other models that illustrate the flow of interactions between users and the system.

Statement of Needs vs **Stakeholder Requirements**

Statement of Needs	Stakeholder Requirements
It tends to be broader and more high-level, outlining the general objectives and outcomes desired The stakeholders involved in creating the statement of needs may include end-users, decision-makers, subject matter experts, and other relevant parties.	They are more detailed and granular, specifying the features, characteristics, and performance criteria that the end product must meet. Stakeholder requirements are typically gathered from different stakeholders who have a vested interest in the project, including end-users, clients, regulatory bodies, and other relevant parties.



An **abstract model** focuses on representing the essential aspects of the system without getting into the implementation details. It's a conceptual representation of the system's behavior, structure, and interactions.

A **use model**, on the other hand, is more specific and detailed than an abstract model. It focuses on describing how different stakeholders will interact with the system to accomplish their tasks or achieve their goals.

Abstract Model Example (Use Case Diagram):
Consider a banking system where customers can perform various transactions. An abstract model, represented as a use case diagram, might include the following:

- Actors: Customer, Bank Teller, ATM
- Use Cases:
    - Withdraw Money
    - Deposit Money
    - Transfer Funds
    - Check Account Balance
    - Update Personal Information

In this abstract model, the focus is on representing the high-level functionality of the banking system without detailing the specific steps involved in each use case.

Use Model Example (Use Case Scenario):

For the "Withdraw Money" use case from the abstract model above, a use model might include a detailed scenario like this:

- Actor: Customer
- Goal: To withdraw cash from their account using an ATM
- Steps:
    - The customer inserts their ATM card into the machine.
    - The ATM prompts the customer to enter their PIN.
    - The customer enters their PIN.
    - The ATM validates the PIN and presents options for withdrawal.
    - The customer selects the amount to withdraw.
    - The ATM dispenses the requested amount of cash.
    - The customer takes the cash and their card from the machine.
    - The ATM displays a transaction summary and asks if the customer wants a receipt.
    - The customer selects whether to print a receipt or not.
    - The ATM completes the transaction and returns to the main menu.

This use model provides a detailed step-by-step description of how a specific interaction between the system and the user (customer) unfolds for the "Withdraw Money" use case.

You are a requirement engineer tasked with developing a new mobile banking application for a leading financial institution. The project stakeholders include bank executives, software developers, user interface designers, and end-users.

1. Requirement Elicitation:

•*Scenario*: During the requirement elicitation phase, you conduct interviews with bank executives and customers to gather information about the desired features and functionalities of the mobile banking application.

- *Question*: How would you use abstract models and use models to elicit requirements effectively in this scenario?

Abstract models such as context diagrams can help in understanding the overall scope of the mobile banking system, while **use models** like use case diagrams can represent specific user interactions such as account balance inquiries or fund transfers.

2. Use Case Modeling:

•*Scenario*: You are tasked with developing use cases for the mobile banking application to capture various user interactions and system behaviors.

- *Question*: Provide three examples of use cases for the mobile banking application and explain how they help in understanding user requirements.

Examples of use cases:

Login: Allows users to securely log in to their accounts.

Check Account Balance: Enables users to view their account balances.

Transfer Funds: Facilitates the transfer of funds between accounts.

Use cases help in understanding the specific functionalities required by users and guide the development process.

3. User Interface Design:

•*Scenario*: The user interface design team is working on creating mockups for the mobile banking application.

- *Question*: How can abstract models and use models influence the design of the user interface to ensure a seamless and intuitive user experience?

Abstract models such as wireframes can provide a basic layout of the user interface, while use models like scenario-based design can help in creating user flows that reflect common tasks performed by users.

4. Non-Functional Requirements:

•*Scenario*: The software development team is discussing the non-functional requirements of the mobile banking application, such as security and performance.

- *Question*: Identify three non-functional requirements for the mobile banking application and discuss their impact on the system design.

Examples of non-functional requirements:

Security: The application must use encryption to protect sensitive user data.

Performance: The application should load quickly and respond to user actions promptly.

Scalability: The system should be able to handle a large number of concurrent users without performance degradation.

Non-functional requirements guide system architecture and implementation decisions.

QUIZ

***Scenario*: The mobile banking application is being deployed in multiple countries with diverse languages and cultural norms.**

- *Question*: How can abstract models and use models accommodate the localization and internationalization requirements of the mobile banking application? Provide strategies for ensuring that the application is accessible and user-friendly across different regions.

Abstract Models for Localization and Internationalization:

•**Language Support**: Update the abstract models to include language and locale parameters that define the internationalization requirements of the application. This could involve incorporating language-specific resources and localization files.

•**Cultural Preferences**: Consider cultural preferences such as date and time formats, currency symbols, and address formats in the abstract models. These preferences can vary between regions and may require customization for different locales.

Use Models for Localization and Internationalization:

•**Language Selection**: Create use models that depict the user flow for selecting language preferences within the application. This could include scenarios for language selection during the onboarding process or within user settings.

•**Currency Conversion**: Develop use models that illustrate how currency conversion is handled within the application. This includes scenarios for displaying account balances and conducting transactions in different currencies.

•**Regional Settings**: Capture user scenarios for configuring regional settings such as time zones, date formats, and measurement units. Use models can depict how users can customize these settings based on their location and preferences.

Localization Resources:

•**Translation Management**: Utilize abstract models to identify text strings and user interface elements that require translation. Use models can then be used to document the translation process, including the use of translation management tools and collaboration with localization teams.

•**Cultural Adaptation**: Use abstract models to define cultural adaptation requirements, such as localized images, icons, and graphics. Use models can outline the process for adapting visual elements to align with cultural norms and preferences.

Testing and Quality Assurance:

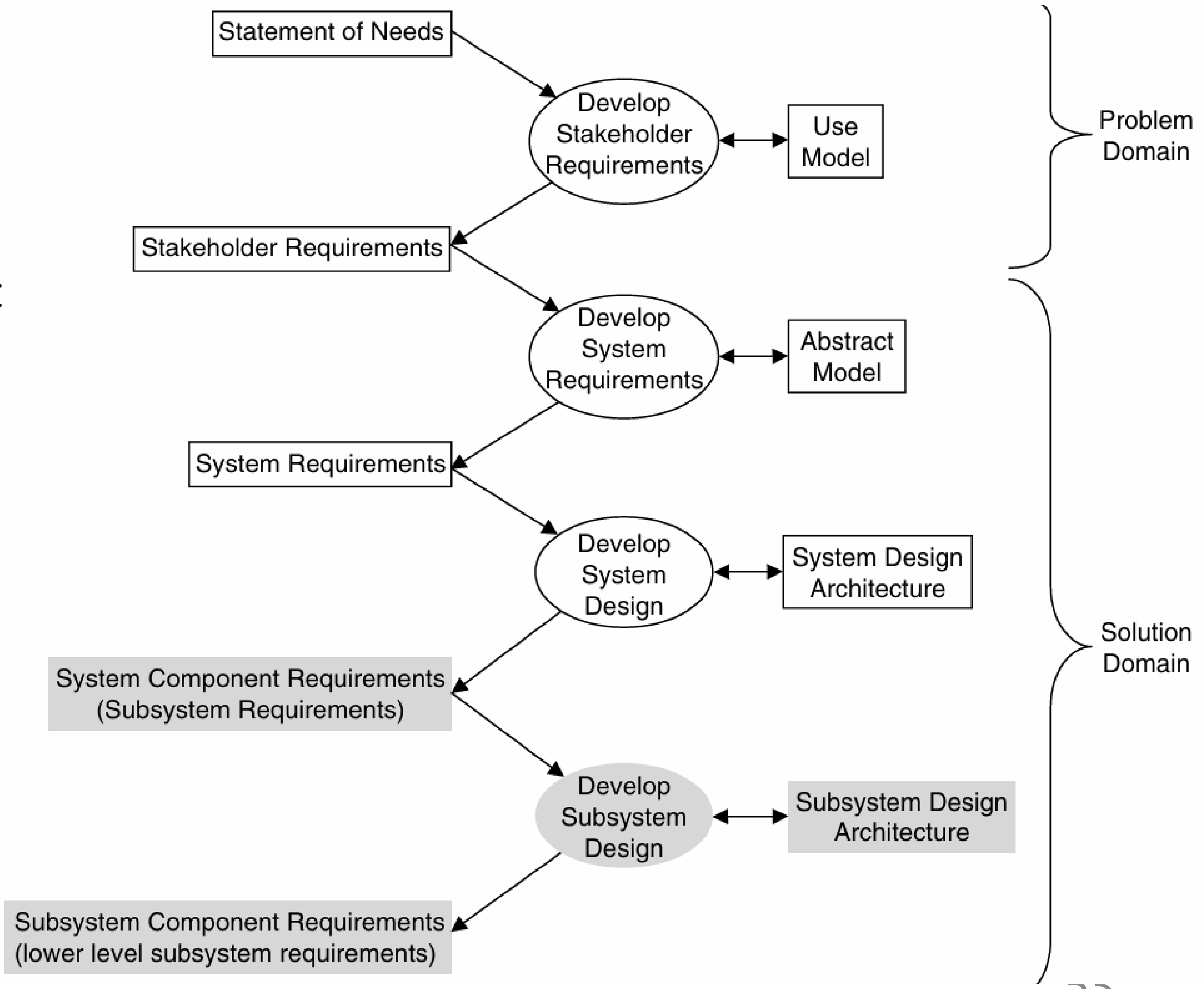
- **Localization Testing**: Develop use models for localization testing, including scenarios for verifying language support, currency formatting, and regional settings across different locales.
- **User Acceptance Testing**: Use models can guide user acceptance testing (UAT) for international users, ensuring that the application meets the needs and expectations of diverse user demographics.

Continuous Improvement:

•**User Feedback**: Use abstract models to capture user feedback related to localization and internationalization. Analyze user feedback and iterate on the localization process to address any issues or discrepancies.

•**Market Analysis**: Utilize abstract models to conduct market analysis and identify opportunities for expansion into new regions. Use models can then be used to define localization requirements for targeted markets.

At all levels from the system components downward, there is multiple concurrent work on requirements at each level. (The grey back ground of the relevant symbols)


Statement of Needs
Develop Stakeholder Requirements
Use Model
Problem Domain
Stakeholder Requirements
Develop System Requirements
Abstract Model
System Requirements
Develop System Design
System Design Architecture
Solution Domain
System Component Requirements (Subsystem Requirements)
Develop Subsystem Design
Subsystem Design Architecture
Subsystem Component Requirements (lower level subsystem requirements)

"Developing sub-system design" in requirements engineering refers to the process of breaking down the overall system into smaller, more manageable subsystems and designing each of these subsystems in detail.
When a system is complex or large, it's often beneficial to divide it into smaller components or subsystems to simplify the design, implementation, and maintenance processes. Each subsystem typically has its own set of requirements, interfaces, and functionalities.

The process of developing sub-system designs involves:

1. **Decomposition**: Breaking down the system into smaller, coherent subsystems based on functional or logical divisions.
2. **Specification**: Defining the requirements and constraints for each subsystem. This involves detailing the functionality, performance, interfaces, and other relevant characteristics.
3. **Design**: Creating detailed designs for each subsystem, which may involve selecting appropriate technologies, algorithms, data structures, and architectural patterns.
4. **Integration**: Planning how the subsystems will interact and integrating them into the overall system architecture.
5. **Testing**: Ensuring that each subsystem functions correctly on its own and also integrates smoothly with other subsystems.

That’s it